Lernen mit Sinix

G. Dischinger / V. Lorenz / G. Maurer

Sinix/Unix

Oldenbourg

Lernen mit Sinix

Eine Reihe von Arbeitsbüchern
für das Selbststudium, herausgegeben von
Prof. Dr. Herbert Kargl und Dipl.-Math. Volkhard Lorenz,
Universität Mainz.

Sinix / Unix

von
Guido Dischinger
Volkhard Lorenz
Gerd Maurer

R. Oldenbourg Verlag München Wien 1991

Die Deutsche Bibliothek – CIP-Titelaufnahme

Dischinger, Guido:
Sinix-Unix / von Guido Dischinger ; Volkhard Lorenz ; Gerd
Maurer. – München ; Wien : Oldenbourg, 1991
(Lernen mit SINIX)
ISBN 3-486-22004-7

NE: Lorenz, Volkhard:; Maurer, Gerd:

Gesamtherstellung: Huber KG, Dießen

ISBN 3-486-22004-7

Inhalt

Vorwort der Herausgeber

Das Betriebssystem **Unix** gewinnt als leistungsfähiges und komfortables Betriebssystem ebenso wie die darauf basierende Standard-Anwendungssoftware wie Textverarbeitung und Tabellenkalkulation immer mehr an Bedeutung. Diesem Wachstum, das sich im Bereich von Personal Computer über Mikrocomputer und Compiler bis zu größeren DV-Anlagen vollzieht, hängt jedoch die Ausbildung im universitären wie im kommerziellen Bereich zur Zeit noch weit hinterher.

Da auf dem europäischen Markt die **Unix**-Mehrplatzsysteme der Firma Siemens Nixdorf Informationssysteme AG mit dem **Unix**-Derivat **Sinix** am weitesten verbreitet sind, basiert die vorliegenden Reihe "Lernen mit **Sinix**" auf diesem Betriebssystem. Bis auf geringfügige Einschränkungen sind die Arbeitsbücher zu den jeweiligen Software-Produkten auch in anderen Systemumgebungen anwendbar.

Grundlage für die Reihe "Lernen mit **Sinix**" waren unsere praktischen Erfahrungen, sowohl in der Durchführung von Tutorials im Rahmen der Grundausbildung in EDV von Studierenden der Wirtschaftswissenschaften an der Universität Mainz als auch in der Schulung von Anwendern im Unternehmen.

Durch diese Arbeitsbuchreihe wird der Leser in die Lage versetzt, sich selbständig in die jeweilige Software einzuarbeiten. Da ein solches Einarbeiten nur durch praktische Arbeit am Bildschirm effizient ist, sind diese Arbeitsbücher auf Übungsbeispielen und Aufgaben aufgebaut, die den Leser schrittweise in den Aufbau und in die Handhabung der jeweiligen Software-Produkte einführen und die direkt am Computer durchgeführt und gelöst werden können.

Zielgruppen für die Arbeitsbuchreihe sind einerseits Studierende, die im Rahmen einer Basisausbildung zur EDV die entsprechende Mikrocomputer-Software erlernen sollen,

und andererseits Anwender in Behörden und Unternehmen, die Software-Produkte dieser Art in ihrer täglichen Arbeit einsetzen werden.

Die Reihe "Lernen mit **Sinix**" soll und kann die jeweiligen Software-Handbücher nicht ersetzen; durch ein ausführliches Register und durch zahlreiche Marginalien können sie dem Anwender jedoch auch als Nachschlagewerke zu dem jeweiligen Software-Produkt dienen.

Dieses Arbeitsbuch gibt eine Einführung in das Betriebssystem, so wie sie für das Verständnis von Standardanwendungssoftware und den Einsatz von Compilern notwendig ist. Darüber hinaus bietet es dem Systemverwalter mit zahlreichen Beispielen und Hinweisen eine wertvolle Unterstützung für seine tägliche Arbeit.

In der Reihe "Lernen mit **Sinix**" sind bereits erschienen die Arbeitsbücher zu HIT (Textverarbeitungssystem), SIPLAN (Tabellenkalkulationsprogramm) und INFORMIX (Datenbanksystem).

Mainz, im August 1991

Herbert Kargl Volkhard Lorenz

Einleitung

Unix[1] ist ein Betriebssystem, das sich auf Rechnern aller Größenklassen - vom PC bis zum Großrechner - zunehmender Beliebtheit erfreut. Dieses Buch soll Sie zum Kennenlernen und Anwenden von Sinix[2], dem Unix der Siemens Nixdorf Informationssysteme AG, anleiten. Wenn Sie also auf Betriebssystemebene arbeiten (wollen), Aufgaben der Systemadministration wahrnehmen (möchten) oder als Studierende(r) Ihre Grundkenntnisse in Datenverarbeitung ausbauen wollen, dann hoffen wir, Ihnen mit diesem Buch das Richtige zu bieten, um sich die erwünschten Kenntnisse im Selbststudium anzueignen.

Erfahrungen im Umgang mit dem Computer sind dabei weder notwendig noch hinderlich. Allerdings sollten Sie gegebenenfalls einen netten Mitmenschen ansprechen können, der Ihnen zeigt, wie Zentraleinheit, Bildschirm und Drucker eingeschaltet werden, wie Sie eine Diskette einlegen usw. (Das kann Ihnen jemand "live" viel besser und schneller erklären als ein Buch.) Bitte besorgen Sie sich auch eine sogenannte Benutzerkennung mit Shell-Erlaubnis.

Falls Sie schon etwas fortgeschritten in der Anwendung von Unix oder speziell Sinix sind, kann Ihnen dieser Band als Nachschlagewerk dienen: Die Kommandos sind innerhalb von Sachgruppen alphabetisch geordnet und noch einmal im Anhang erschlossen. Darüber hinaus wird die gezielte Nutzung durch Marginalien und ein Register erleichtert.

---

1) Unix ist ein Warenzeichen von UNIX System Laboratories, Inc.

2) Sinix ist die SNI-Version des Softwareproduktes Xenix und enthält Teile, die dem Copyright der Microsoft Corporation (1982) unterliegen; die übrigen Teile unterliegen dem Copyright der Siemens Nixdorf Informationssysteme AG. Xenix ist ein Warenzeichen der Microsoft Corporation und ist aus Unix-Systemen unter Lizenz von AT&T entstanden.

Das Buch ist praxisnah konzipiert: Sie lernen Sinix Schritt für Schritt kennen. Nach der Lektüre der ersten beiden Kapitel - also nach etwa 20 Seiten - sollten Sie die Übungsaufgaben in Kapitel 6 bearbeiten. Dort werden Sie, wo immer es nötig ist, auf die entsprechenden Stellen der "Nachschlagekapitel" 3 und 4 verwiesen. Hier finden Sie ausführliche, wirklichkeitsnahe Beispiele und Erläuterungen zu den Kommandos. Selbstverständlich stehen in Kapitel 6 auch die Lösungen zu den (meisten) Aufgaben.

Besonders hilfreich für Systemadministratoren ist das Kapitel 5, in dem wir konkrete Problemlösungen für den laufenden Betrieb vorstellen. Die diesem Buch beiliegende Diskette enthält neben den Beispieldaten für die Übungsaufgaben auch die wichtigsten behandelten Shell-Prozeduren sowie Demonstrations-Prozeduren.

Diesem Buch liegt die Sinix-Version 5.22 zugrunde; Informationen über die Version 5.40, soweit verfügbar, haben wir selbstverständlich aufgenommen. Herrn Stephan Bellarz danken wir an dieser Stelle für die Durchsicht des Typoskripts und das Durcharbeiten der Übungen.

Mainz, im Juni 1991

Guido Dischinger
Volkhard Lorenz
Gerd Maurer

1. Grundsätzliche Überlegungen zum Betriebssystem Sinix

1.1 Betriebssysteme, insbesondere Unix

Betriebssysteme

Das Betriebssystem hat die Aufgabe, Betriebsmittel (Prozessoren, Hauptspeicher und periphere Einheiten wie Drukker, Festplatten und Diskettenlaufwerke), Programme und Datenablagestrukturen zu verwalten und die dazugehörigen Arbeitsabläufe zu steuern. Es ist der "Manager" des gesamten Systems, ohne den "nichts läuft", denn erst, wenn das Betriebssystem in den Hauptspeicher geladen worden ist, kann der Computer mit Anwendungsprogrammen arbeiten.

Unix

Dem Betriebssystem **Unix** wird aufgrund seiner besonderen Merkmale oft eine glänzende Zukunft vorausgesagt. Unix wurde bereits seit 1969 in den Bell Laboratories, einem von Western Electric und AT&T gegründeten Forschungszentrum, entwickelt. Eine standardisierte Version wird von AT&T erst seit Anfang 1984 vertrieben. Aus Unix-Systemen entwickelte Microsoft unter Lizenz von AT&T eine Unix-Nachbildung: Xenix. **Sinix** ist die teilweise dem Copyright von Microsoft unterliegende SNI-Version von Xenix. Sinix/ Unix hat unter anderem folgende Merkmale:

Sinix

Merkmale

- Mehrere Benutzer können unabhängig voneinander arbeiten (Multi-User-System);
- es können mehrere Prozesse (im Sinne von Aufgaben) vom Rechner gleichzeitig bearbeitet werden (Multi-Tasking);
- Datensicherheit ist durch paßwortgeschützte Kennungen und Möglichkeiten zur Manipulation von Datenzugriffsrechten gewährleistet;
- baumstrukturiertes (hierarchisches) Dateisystem;
- schneller Kommando-Interpreter ("Shell"), der auch als Programmiersprache nutzbar ist;
- vielfältige Kombinationsmöglichkeiten der Kommandos;
- C-Programme sind von Unix-System zu Unix-System portierbar, d. h. übertragbar und lauffähig;
- die "Unix-Pipe" (direkte Programmkommunikation).

Unix ist ein interaktives (dialogorientiertes) Timesharing-System, d. h., die Systemressourcen (z. B. die Zeit der Inanspruchnahme des Zentralprozessors und des Hauptspeichers) werden gleichmäßig auf alle Programme bzw. Benutzer verteilt. Man bezeichnet den dahinterstehenden Algorithmus auch als "Zeitscheibensystem", da den Abnehmern in schneller Folge reihum jeweils eine bestimmte Nutzungszeitspanne zugeteilt wird. Dies geschieht so schnell, daß der einzelne Anwender in der Regel nichts davon bemerkt.

1.2 Was ist die "Shell"?

Definition der Shell

Die Shell ist die Schnittstelle zwischen Benutzer und Betriebssystem. Sie verbindet den Benutzer direkt mit dem Kern des Systems, der das eigentliche Betriebssystem ausmacht. Er wird beim "Hochfahren" (Einschalten) des Rechners in den Hauptspeicher geladen und besteht aus den vier Komponenten:

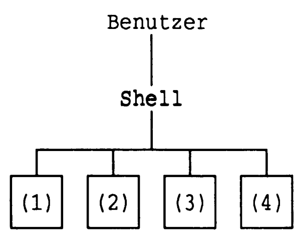
(1) **Dateiverwaltungssystem**
(2) **Prozeßverwaltung**
(3) **Speicherverwaltung**
(4) **Ein-/Ausgabesystem**

Schematisch sieht dies folgendermaßen aus:

Kommando-Eingabe

Kommando-Interpreter

Systemkern



Kommando-Interpreter

"Kommando-Interpreter" bedeutet, daß Kommandos an den Kern des Betriebssystems für diesen verständlich zur Verarbeitung weitergegeben (interpretiert) und die Ergebnisse dieser Kommunikation ausgegeben werden (z. B. am Terminal). Kommandos sind Befehle (Programme), durch die der Benutzer dem Computer genau mitteilt, was zu tun ist.

Aufgrund der Schnelligkeit des Kommando-Interpreters und der vielfältigen Kommandos zur Ablaufsteuerung (Schleifen, Abfragen usw.) kann die Shell auch wie eine Programmiersprache verwendet werden: Unix bietet die Möglichkeit, Befehle zusammenzustellen und abzulegen, so daß sie wie ein Programm "ausgeführt" werden können. Man spricht dann von einer "Shell-Prozedur". Solche Prozeduren sind vor allem bei der Lösung komplexer Aufgaben relevant.

Um Mißverständnissen vorzubeugen, muß die doppelte Verwendung des Begriffes "Shell" verdeutlicht werden:
"Die Shell" ist die Bezeichnung für die Kommando-Ebene des Systems an sich, für den Kommando-**Interpreter**. Sie ist nichts anderes als ein Programm, das Benutzereingaben an das eigentliche Betriebssystem weitergibt. **"Eine Shell"** dagegen ist eine Ablaufumgebung für ein einzelnes Kommando, die direkt nach dem Aufruf des Kommandos erzeugt und nach der Ausführung des Befehls wieder beendet wird.

"die Shell"

"eine Shell"

Die Ausführung eines Programms bzw. Kommandos unter dieser Ablaufumgebung wird als "Prozeß" bezeichnet. Jeder Prozeß wird vom Betriebssystem mit einer eindeutigen Prozeßnummer versehen, mit der man während des Programmablaufs auf ihn Bezug nehmen kann. Die meisten Ihrer Prozeduren werden spätestens dann beendet (abgebrochen), wenn Sie Ihr Terminal abschalten. Eine Ausnahme ist z. B. der Login-Prozeß.

Prozeß

Prozeßnummer

Das Prozeßkonzept basiert auf einer Hierarchiestruktur: Beim Systemstart wird eine Ursprungs-Shell erzeugt, die der "Vaterprozeß" aller im weiteren Verlauf erzeugten "Kind"-Prozesse, der sog. **Subshells**, ist. Man kann aus jeder Shell theoretisch beliebig viele Subshells erzeugen, die dann in eine Prozeßliste eingetragen werden. Das macht sich auch beim praktischen Arbeiten bemerkbar: Man kann eine Shell erst ordnungsgemäß beenden, wenn alle ihre Subshells ebenfalls beendet sind. Wird eine Shell "gewaltsam" aus der Prozeßliste gelöscht, so "sterben" automatisch auch ihre Subshells.

Subshell

1.3 Vor- und Nachteile der Unix-Shell

Der Einsteiger (und nicht nur dieser) fragt sich oft, warum er eine komplexe Kommandosprache lernen soll, wenn es doch schließlich auch einfachere Menüoberflächen gibt. Die Shell bietet gegenüber menügesteuerten Benutzeroberflächen mehrere Vorteile:

Vorteile

- Menüsteuerungen sind meistens von System zu System verschieden und sogar beim gleichen Hersteller nicht dauerhaft einheitlich. Die Kommando-Ebene ist jedoch bis auf wenige Unterschiede bei allen Unix-Systemen sehr ähnlich (z. B. dem "X/OPEN"-Standard folgend).

- Die Shell ist ein umfassenderes, mächtigeres, flexibleres und schnelleres Instrument, als es Menüs jemals sein können. Vor allem das durch Shell-Prozeduren gegebene hohe Automatisierungspotential ist ein wichtiger Vorteil.

- "Trouble-Shooting" wird hauptsächlich in der Shell durchgeführt.

- Erfahrungsgemäß kommt jeder Benutzer einmal an den Punkt, an dem Menükenntnisse nicht ausreichen.

Nachteile

Mehrere Nachteile der Shell sind jedoch nicht von der Hand zu weisen:

- Die Oberfläche ist völlig auf Schnelligkeit und Flexibilität ausgerichtet: Die kurze und abgehackte Kommandoschreibweise, häufig fehlende Rückfragen und wenig verständliche Fehlermeldungen bewirken, daß vor allem Anfängerfehler kaum verhindert werden.

- Die Oberfläche der Shell ist im allgemeinen nicht besonders komfortabel, was sich z. B. im Fehlen von nennenswerten Online-Hilfe-Systemen dokumentiert.

- Allzu umfassende und benutzerunfreundlich strukturierte Dokumentationen machen Unix etwas unübersichtlich. Hier setzt die vorliegende Veröffentlichung an.

Die Nachteile können jedoch zum Vorteil gewendet werden. Beispielsweise gibt es mittlerweile auch für die Shell Hilfe-Systeme, und es existieren auch komfortablere Oberflächen für die Kommando-Ebene.

2. Grundlagen der Sinix-Shell

2.1 Kommando-Eingabe in der Shell

Kennung

Login

Kennwort

Um überhaupt Kommandos eingeben zu können, müssen Sie (vom Systemverwalter) eine Kennung erhalten haben, unter der Sie sich beim Betriebssystem anmelden können ("Login"). Sie führen den Login durch, indem Sie (bei laufender Zentraleinheit) ein Terminal einschalten, Ihre Benutzerkennung eingeben und auf die Taste <RETURN> drücken (Taste mit dem Symbol "<—┘"). Das System fragt Sie dann nach Ihrem Kennwort, das Sie nun eingeben müssen. Die Eingabe geschieht aus Datensicherheitsgründen unsichtbar. Korrigieren können Sie zeichenweise rückwärts mit der <BACKSPACE>-Taste (Symbol: ein nach links zeigender Pfeil mit einem Kreuz innen). Wenn das Kennwort akzeptiert wurde, wird das für Ihre Kennung vom Systemverwalter eingetragene Startprogramm aufgerufen (im allgemeinen ein Menü- oder Shell-Prozeß).

Groß-/Kleinschreibung

An dieser Stelle ist darauf hinzuweisen, daß die Shell **immer** zwischen Groß- und Kleinschreibung unterscheidet (also bei Namen von Kennungen, Paßwörtern, Namen von Dokumenten und Kommandos).

Kennungsklassen

Die Benutzerkennungen im System sind verschiedenen "Klassen" zugeteilt, die unter anderem die standardmäßige Arbeitsumgebung der Benutzer festlegen (z. B. Menü oder Shell). Benutzer der Klasse "expert" befinden sich sofort nach dem Login in der Shell. Benutzer der Klasse "prog" erreichen aus dem Standard-Menüsystem die Shell, indem sie ein Ausrufezeichen eingeben und die <RETURN>-Taste drükken. Kennungen, die der Klasse "collage" angehören, wählen unter COLLAGE die Anwendung "Shell".

Bereitzeichen

Der Interpreter ist bereit, wenn das sogenannte Bereitzeichen (= Prompt, Eingabeaufforderung) erscheint: ein Dollarzeichen ("$") und ein Leerzeichen für normale Benutzer. Falls ein Kommando über mehrere Zeilen geschrieben werden

muß, wird am Anfang der Folgezeilen das "sekundäre" Bereitzeichen ">" ausgegeben. Beim Systemverwalter (die Kennungen "root" und "admin") ist das Bereitzeichen ein Kreuz ("#"). Dem Systemadministrator sind im System praktisch keine Grenzen gesetzt, man sollte deshalb bei der Weitergabe der betreffenden Kennwörter sehr vorsichtig sein. Systemverwalter

Sobald der Prompt erscheint, tippen Sie eine Kommandozeile ein und "schicken" sie mit <RETURN> ab. Hier ein Beispiel (die Benutzereingaben sind unterlegt): Kommando-Eingabe

Beispiel

```
$ echo "Ich bin in der Shell"          (Kommando-Eingabe)
Ich bin in der Shell                   (Ausgabe)
$                                      (Bereitzeichen)
```

Wenn der Befehl nicht ausgeführt werden kann (etwa wegen falscher Schreibweise oder zu langer Kommandozeile), gibt Sinix eine Fehlermeldung aus (z. B. "nicht gefunden"), bricht ab und gibt wieder das Bereitzeichen aus. Der Befehl wird ebenfalls abgebrochen, wenn die Kommandozeile deutlich länger als 9000 Zeichen ist.

Korrigieren können Sie zeichenweise rückwärts mit der **<BACKSPACE>**-Taste, **nicht** aber mit den Pfeiltasten. Eingabekorrektur

Mit der <ENDE>-Taste beenden Sie die Shell wieder. Shell beenden

WICHTIGER HINWEIS:
Jedes Kommando kann während seiner Laufzeit mit der Taste **<DEL>** (DELete = Löschen) wieder abgebrochen werden, soweit diese Taste für die aktuelle Shell nicht gesperrt wurde. Sie können die <DEL>-Taste auch drücken, wenn Sie sich in einer Kommandozeile so sehr verschrieben haben, daß eine Korrektur mit <BACKSPACE> nicht mehr sinnvoll ist. Sie erhalten sofort ein neues Bereitzeichen. Kommando-Abbruch

<u>Zur Syntax der Kommandos:</u>

Die "Syntax" eines Kommandos ist sozusagen seine in allgemeine Regeln gefaßte Schreibweise. In diesem Buch sollen

folgende Konventionen gelten:

Syntax

Kommando (Schalter) ([Alternativen]) (Objekt(e))

In der Shell ist jedes einzelne Zeichen wichtig. Vergessen Sie also nicht, die Komponenten der Kommandozeile mit jeweils mindestens einem Leerzeichen voneinander zu trennen! Die Klammern sollen nicht eingegeben werden, sondern haben wie auch die anderen Angaben eine symbolische Bedeutung:

Kommando:	Sinix-Befehl (z. B. "**ls**")
Schalter:	Schalter bewirken eine spezielle Ausprägung des Kommandos (z. B. "ls **-d**").
Objekt(e):	Sie bezeichnen das, worauf sich das Kommando bezieht (z. B. "ls -l **ORDNER4**").
Runde Klammern:	Die Angaben in runden Klammern sind nicht zwingend vorgeschrieben (optional; z. B. kann man "ls **-a**" oder auch nur "ls" eingeben); "(...)" würde bedeuten, daß das Argument unmittelbar davor (z. B. ein Dateiname) mehrfach vorkommen darf.
Eckige Klammern:	Die Werte sind alternativ (z. B. kann man nur entweder "far **c**" oder "far **x**" angeben, "far cx" wäre syntaktisch falsch, vgl. S. 66).

Optionen

Argumente

Man bezeichnet Schalter auch als "Optionen" und sämtliche Werte hinter dem Namen des Kommandos (also Schalter und Bezeichnungen) als "Argumente".

Beispiel

Als Beispiel für die Anwendung der Kommandosyntax soll das "ls"-Kommando dienen: **ls (-adlRx) (Pfade)**

"ls" ist das eigentliche Kommando (probieren Sie!). Wenn man es eingibt, werden die Einträge (relative Pfadnamen) des aktuellen Dateiverzeichnisses angezeigt. Sind keine vorhanden, wird nur der Prompt ausgegeben, da das Kommando seine Funktion ja ordnungsgemäß erfüllt hat. Wenn auch die

Verzeichniseinträge, deren Namen mit einem Punkt beginnen, angezeigt werden sollen (also z. B. ".profile"), muß man den Schalter "-a" hinzufügen: "ls -a". Um sich zusätzlich über die Zurgriffsrechte sämtlicher Einträge zu informieren (Schalter "-l"), gibt man "ls -al" ein. Wenn man die Zugriffsrechte eines bestimmten Pfades (z. B. das Dokument "/etc/passwd") prüfen möchte, so setzt man den Befehl "ls -l /etc/passwd" ab. Der Befehl "ls" ist im Abschnitt 3.2 vollständig erklärt.

2.2 Ablaufumgebungen bzw. Universen in Sinix V5.xx

In den Sinix-Versionen ab V5.0 unterscheidet man verschiedene globale **Ablaufumgebungen** (**Universen**), die herstellerspezifischen Unix-Versionen nachgebildet sind. Jedes Software-Produkt ist mindestens **einem** bestimmten Universum zugeordnet, z. B. läuft **HIT V4.0** nur im AT&T-Universum ab. Bei **INFORMIX** V2.1 dagegen existieren Versionen für das Siemens- und für das AT&T-Universum. — Universen

Viele Kommandos existieren in allen Universen. Es bestehen generell nur geringe Unterschiede (z. B. bei der Syntax):

Allgemeine Syntax für den Aufruf eines Universums: — Universen aufrufen

Universumsaufruf (Kommando)

Universumsaufrufe sind vor allem die folgenden Befehle:
1. **sie**: Siemens-Universum (Sinix V2.1 der Firma Siemens);
2. **att**: AT&T-Universum (Unix System V der Firma AT&T);

Es gibt weiterhin das Universum der University of California at Berkeley (Aufruf mit dem Befehl "ucb"), das aber in diesem Buch nicht behandelt wird.

Wenn **"Kommando" angegeben** ist, wird der Befehl im jeweiligen Universum ausgeführt. Danach kehrt Sinix wieder in das ursprüngliche (vor dem Aufruf) zurück. **Ohne "Kommando"** wird der für die aufrufende Kennung beim Login aufgerufene Prozeß gestartet (z. B. eine Shell oder das Menüsystem).

Wenn Befehle spezifisch für eine Version oder ein Universum sind, wird dies in den Kommandobeschreibungen von Kapitel 3 und 4 gekennzeichnet, indem über dem Kasten, der die Syntax enthält, Universums- und Versionskürzel hinzugefügt werden. Da das AT&T-Universum den gängigen Unix-Standard abbildet und da ab Sinix V5.4 das Siemens-Universum gesondert hinzugekauft werden muß, beziehen sich im folgenden - soweit nichts anderes angegeben ist - alle Aussagen auf das immer vorhandene AT&T-Universum.

2.3 Das Sinix-Dateisystem

2.3.1 Aufbau des Dateisystems

Dateisystem

Jedes Betriebssystem besitzt ein Dateisystem - ein "Ablagesystem" -, das eine Menge von Daten nach bestimmten Regeln anordnet. Das Dateisystem in Sinix besteht aus einer

Hierarchie

hierarchischen Baumstruktur mit Dateiverzeichnissen (Synonyme: Ordner, Archive, Kataloge, Directories; Abkürzung:

DVZ

DVZ), in denen wiederum DVZ abgelegt werden können, und Dateien (Synonym: Dokumente), die die eigentlichen zu verarbeitenden Daten enthalten, z. B. einen Geschäftsbrief.

root: /

Das "oberste" DVZ, das Ursprungsdateiverzeichnis, heißt **"root"** ("Wurzel") und wird auf der Kommando-Ebene mit dem Zeichen "/" (Schrägstrich, Slash) angesprochen.

Beispiel: /usr/gast/otto

Der erste "/" ist das Ursprungs-Dateiverzeichnis, die "root". Danach trennt das Zeichen "/" (Schrägstrich, Slash) die verschiedenen Komponenten des Pfadnamens: "usr" ist das folgende Dateiverzeichnis, das zweite "/" trennt

die nächste Komponente des Pfadnamens ab. "gast" ist schließlich das Verzeichnis, in dem sämtliche Dateiverzeichnisse und Dateien des Benutzers "gast" abgelegt werden. "otto" kann eine Datei innerhalb des Verzeichnisses "gast" sein, aber auch ein weiteres Dateiverzeichnis.

Jede Kennung im System besitzt ein "Stammverzeichnis", das standardmäßig nur von dieser Kennung zur Ablage von Daten benutzt werden kann (synonym: "Heimatkatalog", Home-Directory). Beim Benutzer "gast" ist dies z. B. das Dateiverzeichnis "/usr/gast" (der Name des Stammverzeichnisses stimmt mit dem Namen der Kennung überein). Wenn Sie aus dem Menüsystem in die Shell wechseln und den Befehl "pwd" (Anzeige des aktuellen Verzeichnisses) eingeben, wird normalerweise der Pfadname Ihres Stammverzeichnisses ausgegeben.

Stammverzeichnis

pwd

Jeder Benutzer sollte aus Gründen der Performance und der Übersichtlichkeit seine Daten sinnvoll und platzsparend organisieren. Wenn man zu viele Einträge (z. B. mehr als 100) in ein DVZ ablegt, sinkt die Performance (die Schnelligkeit des Systems z. B. beim Suchen von Dateinamen), und die Übersichtlichkeit leidet (das Anzeigen aller Einträge erstreckt sich über zu viele Bildschirmseiten).

Individuelles Organisieren

Der Speicherplatz auf der Festplatte wird in mindestens zwei Bereiche ("**Partitionen**") aufgeteilt: den "root"-Bereich (nicht zu verwechseln mit dem Ursprungsdateiverzeichnis) und den "user"-Bereich. Der "root"-Bereich enthält das Betriebssystem, während im "user"-Bereich sämtliche Benutzerdaten und Anwendungssoftware abgelegt werden. Der "user"-Bereich kann wiederum in weitere Bereiche aufgeteilt sein, die logisch in Form von Dateiverzeichnissen vorliegen. Wie diese Dateiverzeichnisse heißen, ist von der Benennung beim Einrichten des Systems abhängig (z. B. "/usr" oder "/usr1"). Die Einteilung in mehrere Subsysteme wird hauptsächlich aus Gründen der Performance und der Systempflege durchgeführt.

Partitionen

2.3.2 Absolute und relative Pfadnamen

Bei der Angabe von Dateien und Dateiverzeichnissen ist zwischen absoluten und relativen Pfadnamen zu unterscheiden:

absoluter Pfadname

Absolut: Angabe mit allen übergeordneten Dateiverzeichnissen (ab der "root")
Beispiel: /usr1/gast/ordner2/datei1

relativer Pfadname

Relativ: Angabe des Pfadnamens relativ zum aktuellen Dateiverzeichnis
Beispiel: ordner2/datei1

"." und ".."

Jedes Dateiverzeichnis enthält mindestens zwei symbolische Einträge, die gleichzeitig als relative Pfadnamen aufgefaßt werden können: "." und "..".
".." bezeichnet immer das übergeordnete Dateiverzeichnis (im folgenden Beispiel "/usr1"), während "." für das aktuelle Dateiverzeichnis steht, in dem man sich gerade befindet ("working directory"). Diese "Abkürzungen" können auch in Pfadnamen verwendet werden (z. B. "../Z3").

Beispiele

Beispiele:

Gehen wir von folgender Struktur aus:

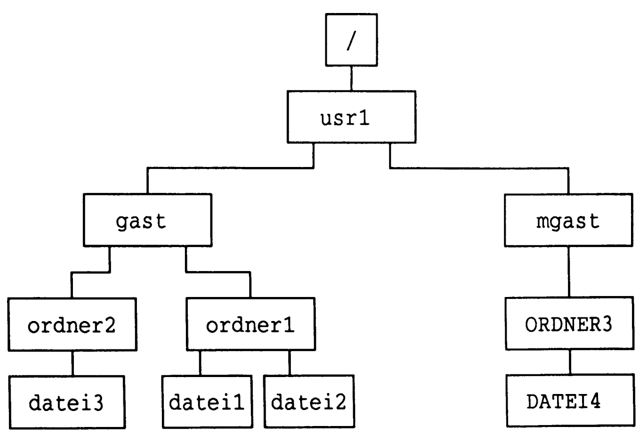


Abb. 2.1: Absolute und relative Pfadnamen

Absolute Pfadnamen der obigen Dokumente bzw. Dateien:

/usr1/gast/ordner1/datei1
/usr1/gast/ordner1/datei2
/usr1/gast/ordner2/datei3
/usr1/mgast/ORDNER3/DATEI4

Relative Pfadnamen:

Befindet man sich im Dateiverzeichnis "/usr1/gast" (vgl. Abb. 2.1), so spricht man z. B.

/usr1/gast/ordner1/datei1 mit ordner1/datei1 an;
/usr1/gast/ordner2 mit ordner2 an;
/usr1/mgast/ORDNER3/DATEI4 mit ../mgast/ORDNER3/DATEI4 an.

cd

DVZ wechseln

Man "wandert" im Dateisystem mit dem Kommando "cd" (change directory). Wenn Sie von einer beliebigen Stelle des Dateisystems in das Verzeichnis "/usr1/gast" wechseln wollen, müssen Sie den Befehl "cd /usr1/gast" eingeben. Wenn Sie von dort (s. Abb. 2.1) nach "/usr1/mgast" wechseln wollen, können Sie dies sowohl absolut ("cd /usr1/mgast") als auch relativ angeben ("cd ../mgast"). Im letzteren Fall "klettern" Sie in einem Schritt einen Ast nach oben (nach "/usr1") und "steigen" dann nach unten auf den Ast "mgast". In zwei getrennten Schritten wechseln Sie, wenn Sie zuerst "cd .." und dann "cd mgast" eingeben.

2.3.3 Namen von Einträgen und Arten von Dateien

Definition "Datei"

Dateien sind logische Einheiten auf einem Datenträger (hier die Festplatte), die die zu verarbeitenden Daten enthalten. Der normale Benutzer benötigt nur den Dateinamen, um mit den enthaltenen Daten arbeiten zu können (entsprechende Zugriffsrechte vorausgesetzt). Die Namen von Einträgen sollen nicht länger als 14 Zeichen sein. (Ab Version 5.40 können die Dateinamen bis zu 255 Zeichen lang sein.) Es ist empfehlenswert, aus Gründen der Kompatibilität mit Anwendungsprogrammen Sonderzeichen (wie z. B. "ö", "]","'") zu vermeiden. Es gibt in Unix verschiedene Arten von Dateien:

Namenskonventionen

Lesbare Dateien

1. "Lesbare" Dateien

Eine typische "Klartext"-Datei ist die Datei "/etc/passwd". Sie enthält sämtliche Kennungen und die zugehörigen Daten (das verschlüsselte Kennwort, den Pfad des Stammverzeichnisses usw.). Mit dem Befehl "pg /etc/passwd" können Sie diese Datei lesen (drücken Sie <RETURN>, bis das Bereitzeichen wieder erscheint), d. h. am Bildschirm anzeigen lassen. "pg" zeigt jedoch nur Dateien ohne besondere (meist unsichtbare) Steuerzeichen korrekt an (sogenannte ASCII-Dateien). Zum Lesen der Dokumente des Textsystems HIT muß man allerdings andere Kommandos verwenden. Trotzdem sind auch derartige Dateien prinzipiell lesbar (und sei es auch nur für bestimmte Software-Systeme wie eben z. B. HIT).

2. Binärdateien

Binärdateien

Programme sind Anweisungen für den Computer. Sie liegen oft in Form von Binärdateien vor. "Binär" heißen diese Dateien, weil sie im für den Computer direkt verständlichen Maschinencode (Kombinationen der Zeichen "0" und "1") abgefaßt sind. Beispiele sind die Kommandos von Unix selbst, die nichts anderes als in Maschinencode umgesetzte ("kompilierte") Programme in der Sprache "C" sind. Sie sind nicht direkt lesbar.

3. Gerätedateien ("Devices")

Gerätedateien

Es ist ein Kennzeichen von Unix, daß periphere Geräte mit Hilfe von besonderen Dateien verwaltet werden. Diese Dateien enthalten besondere (Zugriffs-)Einstellungen, die automatisch vom Systemkern in Hardware-Befehle umgewandelt werden. Sie geben also an, wie welcher Steckeranschluß an der Zentraleinheit benutzt wird und welche Form die ausgetauschten Daten haben sollen. Wird z. B. die Datei "/dev/rts0" angesprochen, greift Sinix auf das Magnetbandkassettenlaufwerk zu und spult die Kassette zunächst einmal an den Bandanfang zurück. Die Gerätedateien sind also die Software-Schnittstellen zur Hardware.

Gerätedateien sollten (vom Systemverwalter) nur in dem Dateiverzeichnis "/dev" abgelegt werden.

Hier einige Beispiele:

Konkrete "Geräte"

Gerätedateien	**Zuständiges "Gerät"**
/dev/fl2	Diskettenlaufwerk ("fl" für "floppy")
/dev/rts0	Magnetbandkassettenlaufwerk (V5.2x)
/dev/tape	Magnetbandkassettenlaufwerk (V5.40)
/dev/mt0	Magnetbandlaufwerk
/dev/lp9001-D01	Drucker, hier des Typs "9001" (Tintenstrahldrucker) am Steckplatz "D01"
/dev/tty00	Terminal, hier: "00" = Konsole
/dev/is0h	Festplattenpartition

Diese Gerätedateien finden in einigen der in den Abschnitten 3.2 und 3.3 dargestellten Kommandos Anwendung. Ihre Funktionsweise wird in den Beispielen der jeweiligen Befehle deutlicher, vor allem beim "cpio"-Kommando.

2.4 Sonderfunktionen der Shell (I)

2.4.1 Dateinamengenerierung

Mit den Sonderzeichen *, ? und [] können Dateinamen bzw. Namen von Dateiverzeichnissen **abgekürzt** werden. Dies ist z. B. bei der Anwendung des Kommandos "cd" (Wechsel des Dateiverzeichnisses) sehr hilfreich. Wenn Sie z. B. "cd /usr/g*" eingeben, erzeugt die Shell intern daraus die Kommandozeile "cd /usr/gast" (soweit "gast" der einzige Name unter "/usr" ist, der mit "g" anfängt).

Dateinamengenerierung
*, ?, []

Jedes Sonderzeichen kann alleine stehen, mehrfach vorkommen und mit anderen kombiniert werden. Die Zeichen zur Dateinamengenerierung werden jedoch nicht hinter den Zeichen "<" und ">" interpretiert. Die Sonderzeichen zur Dateinamengenerierung nennt man auch **Joker**. Eine Zeichenfolge, die Joker enthält, heißt **Matchcode**.

Joker

Matchcode

Die Bedeutung der Zeichen:

- `*` Ersetzt beliebige Zeichenfolgen (auch "nichts", also die "leere" Zeichenfolge); alleinstehend gibt der Stern alle Namen des aktuellen Dateiverzeichnisses an, die nicht mit einem Punkt beginnen.
- `?` Ersetzt genau ein Zeichen (nicht aber Leerzeichen).
- `[...]` Eckige Klammern werden immmer in Verbindung mit anderen Matchcodes (meistens "*") verwendet. Sie ermöglichen die Spezifikation von Zeichenfolgen: "[aev]*" bezeichnet z. B. alle Namen, die mit a, e oder v beginnen. Mit Hilfe des Bindestrichs kann man Bereiche angeben: "[a-c]*" steht für alle Namen, die mit a, b oder c beginnen.
- `[!...]` Das Ausrufezeichen negiert die Werte innerhalb der Klammer. Z. B. paßt "[!a]*" zu allen Namen, die nicht mit a beginnen.

Beispiele und Erläuterungen

Man kann die Bedeutung der "Joker" sehr gut anhand der Kommandos "echo" (Bildschirmausgabe) und "ls" nachvollziehen: "echo *" gibt z. B. alle Namen des aktuellen Dateiverzeichnisses aus, die nicht mit einem Punkt beginnen.

Beispiele

Beispiel	Ausgabe
echo *	alle Namen (ohne einen Punkt als erstem Zeichen)
echo ?*	alle Namen mit mindestens einem Zeichen (z. B. "a", "4")
echo .??*	alle Namen, die mit einem Punkt beginnen und mindestens zwei weitere Zeichen aufweisen (z. B. ".profile", ".12")
echo e*a	alle Namen, die mit "e" beginnen und mit "a" enden (z. B. "erika", "ea")
echo *dok*	alle Namen, in denen die Zeichenfolge "dok" vorkommt (z. B.: "doku", "1dok")
echo ????	alle Namen, die aus genau vier Zeichen bestehen (z. B. "susi", "p2.s")

echo [a-k]*	alle Namen, die mit a, b, c ... oder k beginnen (z. B. "aus", "birgit", "k")
echo [!kap]*	alle Namen, die nicht mit k, a oder p beginnen

2.4.2 Der "Pipe"-Mechanismus

Die sogenannte Pipeline ist ein besonderes Kennzeichen von Unix bzw. Sinix. Sie ermöglicht es, Daten von Kommando zu Kommando bequem zu übergeben, ohne daß die Daten in Dateien zwischengespeichert werden müssen (diese temporäre Pufferung und die notwendige Synchronisation beim Datentransfer übernimmt das Betriebssystem). Pipe: ¦

Voraussetzung für die Verbindung von Kommandos mit einer Pipeline ist, daß die Befehle auch standardmäßig ihre Eingaben vom Terminal beziehen und ihre Verarbeitungsergebnisse auf dem Terminal ausgeben.

Das Pipe-Zeichen ist das "¦", auf der deutschen Tastatur ist es das kleine "ö".

Syntax: Kommando1 ¦ Kommando2 (...) Syntax

Normalerweise (ohne Pipe) würde Kommando1 sein Verarbeitungsergebnis am Bildschirm ausgeben. Durch die Verkettung wird die Ausgabe aber direkt an Kommando2 weitergegeben, welches diese Eingabe anstelle einer Eingabe von der Tastatur oder aus einer Datei akzeptiert. Die Leerzeichen vor und nach der Pipe sind nicht notwendig. Ein Kommando kann natürlich auch eine Prozedur sein. In einer Pipeline können theoretisch beliebig viele Kommandos stehen.

Beispiele und Erläuterungen

Eine sehr nützliche Anwendung ist die folgende:

ls ¦ pg

Diese Kommandozeile listet den Inhalt des aktuellen DVZ auf und hält nach jeder Bildschirmseite an, um auf eine Benutzer-Eingabe zu warten. Das ist bei Dateiverzeichnissen hilfreich, die sehr viele Einträge enthalten, wie z. B. "/bin". - Eine andere Anwendung ist folgende:

```
who ¦ wc -l
```

Hier werden die Kennungen der zur Zeit aktiven Benutzer zeilenweise mit dem Kommando "who" aufgelistet, dessen Ausgaben an das Kommando "wc -l" (Zeilen zählen) weitergegeben wird.

2.4.3 Umleitung der Ein- und Ausgabe

Umleitung mit >, <

In Unix bzw. Sinix existieren grundsätzlich drei "Kanäle": die Standard-Eingabe, die Standard-Ausgabe und die Standard-Fehlerausgabe (für Fehlermeldungen). Per Voreinstellung ist normalerweise die Tastatur die Standard-Eingabe; die Standard-Ausgabe sowohl für Verarbeitungsergebnisse als auch für Fehlermeldungen ist normalerweise der Bildschirm (bei einigen Kommandos ist es eine Datei). Mit den Zeichen "<" und ">" können die Kanäle "umgelenkt" werden: "<" leitet die Standard-Eingabe um, während ">" den Standard-Ausgabe-Kanal umdefiniert. (Zur Umlenkung der Standard-Fehlerausgabe schlagen Sie bitte im Handbuch nach.)

Syntax

Syntax: **Kommando (< Datei1) (>(>) Datei2)**

Es kann jeweils nur eine Ein- bzw. Ausgabe-Datei angegeben werden. Die Ausgabedatei wird überschrieben, falls sie vorhanden ist; wenn sie noch nicht existiert, wird sie neu angelegt. Wenn man ">>" statt ">" verwendet, wird an die Ausgabedatei angehängt, sie wird also nicht überschrieben. Mit dem "cat"-Kommando können Sie so Dateien verbinden.

>> Anhängen

Vorsicht: ">..." und ">>..." werden interpretiert, bevor das eigentliche Kommando ausgeführt wird. Der Befehl

```
cat otto susi > otto
```

löscht zunächst "otto" und schreibt dann (die leere Datei) "otto" und "susi" in die neuerstellte Datei "otto". Der Inhalt von "otto" ist also verlorengegangen.

Beispiele und Erläuterungen

Umlenkung der Ausgabe des Kommandos "ls" in die Datei LS:

ls > LS

Die Ausgabe des Kommandos "echo ..." an den Inhalt der im vorigen Beispiel erzeugten Datei "LS" anhängen:

echo "Das war der Inhalt der aktuellen Directory!" >> LS

Beispiele zur Umlenkung der Standard-Eingabe finden Sie u. a. bei den folgenden Kommandos: cpio -i, mail, write.

2.4.4 Hintergrundprozesse

Das Unix- bzw. Sinix-spezifische Prozeßkonzept beinhaltet auch die Möglichkeit, Prozesse "außerhalb" des aktuellen Benutzerprozesses ablaufen zu lassen: Ein Prozeß kann "in den Hintergrund gelegt" werden. Die Shell wartet nicht auf die Beendigung des Prozesses: Er läuft dann unabhängig von der aktuellen Sitzung ab und kann nicht mit der Taste <DEL> abgebrochen werden (hierzu wird das Kommando "kill" verwendet). Ein Hintergrundprozeß wird normalerweise abgebrochen, wenn Sie sich vom System abmelden.

Hintergrundprozesse

Kein <DEL>

&

Syntax

Syntax: **Kommando &**

(Das Leerzeichen ist nicht unbedingt notwendig.)

Beispiele und Erläuterungen

Wenn ein großes Dokument mit dem Kommando "lpr" gedruckt werden soll, kann es zu einer gewissen Wartezeit kommen, so daß es sich lohnt, den Prozeß in den Hintergrund zu legen:

lpr riesendokument &

Es folgt die Ausgabe der Prozeßnummer des Hintergrundprozesses. Sie ist dann in der Prozeßliste (also in der Ausgabe des Kommandos "ps a", vgl. Abschnitt 3.1) zu finden.

Auch beim Kompilieren (Übersetzen) von Programmen ist das Anlegen eines Hintergrundprozesses nützlich.

Man kann den letzten Hintergrundprozeß mit dem folgenden Kommando abbrechen (die Variable "$!" enthält die Prozeßnummer des zuletzt aufgerufenen Hintergrundprozesses):

kill -9 $!

Warnung: Starten Sie möglichst keine interaktiven Programme (z. B. HIT) als Hintergrundprozeß.

2.4.5 Variablen in der Sinix-Shell (I)

Variablen

Als Variablen werden "Daten" bezeichnet, deren Wert innerhalb eines Programms oder eines Prozesses veränderlich ist: Einem Namen wird für den aktuellen Prozeß ein (veränderlicher) Wert zugewiesen, der später wieder unter diesem Namen abgerufen werden kann. In der Shell werden Variablennamen in der Regel mit dem Sonderzeichen "$" eingeleitet (eine Ausnahme bildet die Verwendung in den Befehlen "export" und "read").

$VarName

Wertzuweisung

Eine **Wertzuweisung** erfolgt, indem man in der Shell den Namen der Variablen (ohne "$") dem gewünschten Wert "gleichsetzt". Damit ist eine neue Variable definiert bzw. eine schon existierende mit einem anderen Wert versehen worden. Ohne Angabe von "Wert" wird der vorherige Inhalt der Variable gelöscht.

Syntax: **Variablenname=(Wert)**

Beispiel: Name=Friedrich

echo

Mit dem "echo"-Kommando können Sie den **Wert** der Variablen **ausgeben** lassen:

Eingabe: echo $Name
Ausgabe: Friedrich

1. Standardvariablen

Unter Standardvariablen sollen hier Systemvariablen verstanden werden, die vom System beim Login automatisch definiert werden und die für die Betriebssystemumgebung und/oder für den Ablauf von Software-Produkten notwendig sind. Beispiele für Standardvariablen:

Standard-variablen

HOME Standard-DVZ des Benutzers, z. B. "/usr/gast".

USER Name des Benutzers, z. B. "gast".

PATH Suchpfade, unter denen nach dem Namen eines aufgerufenen Kommandos standardmäßig vom Interpreter gesucht wird, z. B. "/bin:/usr/bin:.:/usr/bin/HIT" (der Doppelpunkt trennt die Pfade voneinander).

DBPATH INFORMIX-Variable, die Pfade enthält, in denen INFORMIX-Datenbanken zu finden sind (z. B. "/usr/gast/DB:/usr1/lis/literatur").

PS1 Enthält das Bereitzeichen (den Prompt).

$? Enthält den Ende-Status des letzten Kommandos (s. Abschnitt 3.7.3).

$* Steht für alle Stellungsparameter (s. Abschnitt 4.3).

$! Enthält die Prozeßnummer (s. Abschnitt 1.2) des zuletzt abgeschickten Hintergrundprozesses.

$# Enthält die Anzahl der Stellungsparameter.

$$ Enthält die Nummer des aktuellen Prozesses (der aktuellen Shell), die im System eindeutig ist.

Beispiel:

Eingabe: echo $USER $DBPATH $$
Ausgabe: gast /usr/gast/DB:/usr/db/alledaten 1228

2. Selbstdefinierte Variablen

Selbstdefinierte Variablen

In der Shell kann man sich (wie oben bereits dargestellt) eigene Variablen definieren, indem man mit dem Gleichheitszeichen ("=") einem Variablennamen einen Wert zuweist. Relevant ist dies vor allem für Prozeduren, in denen bestimmte (variable) Werte immer wieder benötigt werden.

Erweiterung

Variablen werden wie folgt erweitert, ohne daß man den vorherigen Inhalt "überschreibt":

```
Eingabe: echo $VAR
Ausgabe: ErsterInhalt
Eingabe: VAR="${VAR}und ZweiterInhalt"
         echo $VAR
Ausgabe: ErsterInhaltund ZweiterInhalt
```

{ }

Hier wurde der Variablenname "$VAR" mit geschweiften Klammern von der Zeichenkette "und" abgetrennt (sonst hätte die Shell "$VARund" als Variablennamen aufgefaßt). Die doppelten Anführungsstriche werden benötigt, um den Inhalt "ZweiterInhalt" zusammen mit dem Leerzeichen als eine einzige Zeichenkette zu kennzeichnen.

2.4.6 Wichtige Sonderzeichen der Shell

In der Unix-Shell ist jedes einzelne Zeichen wichtig, wie die folgende vereinfachte Auflistung von "Metazeichen" zeigt:

Sonderzeichen

- \ (Gegenschrägstrich, Backslash) entwertet (Sonder-) Zeichen
- / Trennen der Komponenten von Pfadnamen
- ; Trennen von mehreren Kommandos in einer Kommandozeile
- # Keine Aktion (Kommentarzeichen in der Shell)
- () Mehrere Kommandos in einer Zeile zusammenfassen und als Subshell ausführen

Hinweis: Auch "*", "?" und "[]" sind Metazeichen.

Beispiele und Erläuterungen

Wenn man verhindern will, daß der Stern bei der Ausgabe von Zeichenketten mit dem "echo"-Befehl als Sonderzeichen interpretiert wird, muß man jeden "*" mit dem Gegenschrägstrich ("\") entwerten, so daß er nicht mehr als Sonderzeichen zur Dateinamengenerierung interpretiert wird:

```
Eingabe:    echo \*\*\*\* Hallo \*\*\*\*
Ausgabe:    **** Hallo ****
```

Mit dem Zeichen "#" kann man eine Kommentarzeile einfügen, die von der Shell ignoriert (also nicht ausgeführt) werden soll; Kommentarzeilen sollte man z. B. in Shell-Prozeduren reichlich verwenden. Ein Beispiel: #

```
# Diese Zeile wird von der Shell "überlesen".
echo "Die erste Zeile wurde nicht ausgeführt!"
```

Mit dem Semikolon (";") kann man mehrere Kommandos in einer Zeile zusammenfassen. Sie werden dann nacheinander ausgeführt, ohne daß man sie jeweils nach der Ausführung des vorherigen Kommandos eingeben muß. Im folgenden Beispiel wird zunächst der Inhalt des aktuellen Dateiverzeichnisses aufgelistet und dann abgerufen, welche Benutzer gegenwärtig angemeldet sind: ;

```
ls ; who
```

Man kann eine (zu lange) Kommandozeile über zwei Zeilen schreiben, wenn man an das Ende der ersten Zeile den Backslash ("\") setzt. Auf diese Weise entwertet man das sogenannte Zeilenendezeichen, das durch das Drücken der <RETURN>-Taste unsichtbar erzeugt wird. Die Shell sieht deshalb die beiden Zeilen als eine an und führt die Kommandozeile aus, wenn sie ein nicht entwertetes Zeilenendezeichen erhält. Nützlich ist dies, sobald die Kommandozeile zu lang wird oder wenn man sie übersichtlicher gestalten will. (Und analog kann man auch mehr als zwei Zeilen zu einer Kommandozeile zusammenfassen.) \
Im Beispiel werden verschiedene mit "ls" aufzulistende Pfade übersichtlich untereinander notiert. Abschließend wird die Ausgabe mit der Pipeline zur seitenweise Anzeige an den "pg"-Befehl übergeben:

```
ls /usr/menus/app/control/*.scr \
   /etc/herald/*01* \
   /usr1/gast/DOKUMENTE/rechnungen/generell/*1990* \
   /usr2/DB/probe/dbs \
¦ pg
```

Befehle in runden Klammern, also "(" und ")", werden als Subshell ausgeführt, so daß z. B. Verzeichniswechsel die aktuelle Shell nicht betreffen. Eine Anwendung finden Sie in den Beispielen zum "for"-Kommando (s. S. 108). ()

Nun haben Sie einen ersten Einblick in das Betriebssystem Sinix erhalten. Alles weitere lernen Sie am besten durch Übung, so daß wir Ihnen empfehlen, jetzt die Aufgaben in Kapitel 6 zu bearbeiten.

3. Die Sinix-Kommandos

3.1 Benutzer-Umgebung

date

date — **Aktuelles Datum und Uhrzeit anzeigen**

Beispiele und Erläuterungen

Eingabe: date

Es gibt je nach sprachlicher Oberfläche und Universum drei Varianten:

Ausgabe: Thu Jun 20 08:30:15 MEZ 1991
oder
Ausgabe: Di 20.Jun.1991, 08:30:15 MET
oder
Ausgabe: Di 20.Jun.1991, 08:30:15 MEZ

df

df — **Belegung der Plattenbereiche anzeigen**

Die Plattenpartitionen sind in ihrer Größe natürlich begrenzt. Die Belegung der einzelnen Partitionen (Auslastung) wird in KB (Bytes) und in Prozent angezeigt (disk free). Hier ein Beispiel für die Ausgabe:

Beispiel

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/is0a	7443	4982	1716	47%	/
/dev/is0a	135575	49097	72920	40%	/usr
/dev/is0g	73015	64063	1650	97%	/usr1

Im Beispiel wird deutlich, daß der Bereich "/usr" mit 40 % noch lange nicht voll ist, während der Bereich "/usr1" mit 97 % schon im kritischen Bereich ist.
Ab einer Belegung von etwa 85 % sollte der Systemverwalter neuen Platz schaffen (z. B. alte Benutzerdaten löschen).

du -s (DVZ1) (...)	**Größe von DVZ anzeigen**	du

Zeigt die Speicherplatzbelegung eines Dateiverzeichnisses in Blöcken à 512 Bytes an (im Siemens-Universum à 1 KB, also 1024 Bytes): display used blocks.
Wenn kein Dateiverzeichnis angegeben wird, prüft Sinix das aktuelle, in dem Sie sich gerade befinden.

Beispiele und Erläuterungen

Anzeigen, wieviel Speicherplatz "gast" in seinem Stamm-DVZ belegt hat (Voraussetzung ist Leseberechtigung):

Eingabe: sie du -s /usr/gast

Ausgabe: 535 KB /usr/gast

"gast" hat also in seinem Stamm-DVZ 535 KB belegt.

Anzeigen, wie groß die Dateiverzeichnisse "/usr/gast/allgemein" und "/usr/gast/extrem" sind:

Eingabe: att du -s /usr/gast/allgemein /usr/gast/extrem

```
Ausgabe: 10     /usr/gast/allgemein      (ca. 5 KB)
         130    /usr/gast/extrem         (ca. 65 KB)
```

---

att

id	**Ausgabe von User- und Group-ID(entification)**	id

Jedem Benutzer ist ein eindeutiger Kennungsname und eine eindeutige Benutzernummer sowie ein Gruppenname und eine Gruppennummer zugeordnet. Mit diesem Kommando können die Werte abgefragt werden.

Beispiele und Erläuterungen

Eingabe: id

Ausgabe: uid=105(gerd) gid=12(lehrst)

In diesem Beispiel ist die User-Identification "105" bzw. "gerd", die Group-Identification "12" bzw. "lehrst".

---

kill -9

kill -9 Prozeßnummer **Prozesse zerstören**

Benutzen Sie diesen Befehl, wenn ein Prozeß "hängt", d. h., wenn an Ihrem Terminal keine Reaktion mehr erfolgt. Sie müssen dazu an ein anderes Terminal gehen und sich dort unter derselben Kennung anmelden, da Sie nur die eigenen Prozesse zerstören dürfen.
Die Prozeßnummer müssen Sie aus der Ausgabe des "ps au"-Befehls heraussuchen. Normalerweise werden die Kindprozesse bei Zerstörung des Vaterprozesses automatisch abgebrochen.

Der Systemverwalter kann jeden Prozeß zerstören.

Beispiele und Erläuterungen

Betrachten Sie folgende Ausgabe eines "ps a"-Befehls:

```
 PID TT STAT  TIME COMMAND
1668 03 S     0:00 /usr/bin/HIT/fit (= HIT-Menü)
1672 03 S     0:00 hit -h/usr/lib/hit/tmp/FITa01668 -P1668
                                       (= HIT-Editor)
```

Angenommen, der HIT-Prozeß an der Konsole soll zerstört werden. Der HIT-Menü-Prozeß ist der Vater des Editor-Prozesses, da die Prozeßnummer des Menü-Prozesses kleiner ist und somit früher erzeugt wurde. Deshalb wird der Prozeß mit der Nummer 99 "gekillt":

Eingabe: kill -9 99

(Auf dem Bildschirm des defekten Prozesses kommt die Rückmeldung: "99: Prozeß abgebrochen")

Manchmal ist es hilfreich, mit Systemvariablen zu arbeiten: Die Variable "$$" enhält die aktuelle Prozeßnummer, in "$!" ist die Process-Identification des letzten Hintergrundprozesses abgelegt. Den aktuellen Prozeß könnte man also wie folgt zerstören:

kill -9 $$

Zerstören des letzten Hintergrundprozesses:

kill -9 $!

Wenn Sie alle aktuellen Prozesse, die auf Ihrem Terminal aktiv sind (nicht für die Konsole gültig), abstellen wollen, können Sie mit dem folgenden Befehl die gesamte

"Prozeßgruppe" zerstören (**nicht als Systemverwalter!** Die Kennungen "root" und "admin" sind Eigentümer grundlegender Systemprozesse!):

kill -9 0

---

passwd (Kennung) **Kennwort ändern** passwd

Ändert das Kennwort (password) von Kennungen.

"root" bzw. "admin" dürfen das Kennwort aller Benutzer ändern. Sie können auch das Kennwort eines anderen Benutzers ändern, wenn Ihnen sein aktuelles Kennwort bekannt ist (wird vorher abgefragt). Ansonsten darf nur das eigene Kennwort geändert werden, wobei die Angabe der Kennung nicht nötig ist, da per Voreinstellung die aktuelle angenommen wird. Sie müssen das Kennwort zweimal unsichtbar eingeben.

<u>Beispiele und Erläuterungen</u>

Der Benutzer "gast" will sein Kennwort ändern:

Eingabe: passwd

Ausgabe: (Zunächst wird das alte Kennwort abgefragt, dann muß das neue zweimal eingegeben werden.)

---

ps (Schalter) **Prozesse auflisten lassen** ps

Ohne Schalter werden nur die eigenen Prozesse auf dem aktuellen Terminal ausgegeben (process status).

<u>Auswahl der möglichen Schalter:</u>

a : auch die Prozesse anderer Benutzer ausgeben;

u : ausführlichere Information (mit Angabe der Kennung);

x : auch Systemprozesse auflisten;

txx : terminalspezifische Prozesse listen ("xx" = Nummer).

Beispiele und Erläuterungen

Sie müssen Ihre Prozeßnummer kennen, um das "kill"-Kommando anwenden zu können. Wenn Sie einen eigenen Prozeß zerstören wollen, informieren Sie sich mit dem "ps a"-Befehl.

Beispiel für die Ausgabe von "ps ax" (Auszug):

PID	TT	STAT	TIME	COMMAND	(Erläuterung)
0	?	D	0:00	swapper	Systemprozesse
1	?	S	0:08	(init)	
2	?	D	0:00	pagedaemon	
1648	03	I	0:02	sh /usr/menus/sabin/ums	Menü-Shell
1668	03	S	0:00	/usr/bin/HIT/fit	HIT-Menü
1672	03	S	0:00	hit -h/usr/lib/hit/tmp/FITa01668 -P1668	HIT-Editor
1602	04	S	0:01	-sh (sh)	Kommando-Shell
1673	04	R	0:00	ps ax	ps-Prozeß

Unter "PID" stehen die Prozeßnummern, die bei "kill" angegeben werden müssen. Es wird deutlich, daß nur an der Konsole ("00") und an Terminal "04" Prozesse aktiv sind (Spalte "TT").

Als Systemverwalter sollte man vorsichtig sein, da "admin" und "root" **alle** Prozesse abbrechen dürfen:
Zerstören Sie möglichst keine elementaren Systemprozesse!
Systemprozesse laufen über die Konsole und sind mit dem Schalter "x" abzurufen. Wenn man z. B. den Prozeß "0" (steuert das "Swappen", also die Auslagerung von Prozeßseiten aus dem Hauptspeicher auf die Platte) mit "kill" beenden würde, wäre ein Systemabsturz unvermeidlich.

Alle Prozesse des Terminals "01" auflisten:

ps at01

pwd

pwd Pfad des aktuellen Dateiverzeichnisses anzeigen

Dieser Befehl dient zur Orientierung im Dateisystem ("Wo bin ich?"). Ausgegeben wird der absolute Pfadname des aktuellen Dateiverzeichnisses (print working directory).

Beispiele und Erläuterungen

Im Dateiverzeichnis "/usr/gast/Dokumente" eingeben:

Eingabe: pwd

Ausgabe: /usr/gast/Dokumente

---

su (Kennung)	Benutzerkennung in der Shell wechseln	su

Wenn man Aufgaben unter einer anderen Kennung zu erledigen hat, ohne den aktuellen Prozeß durch einen Logout (Abmeldung) beenden zu wollen, wechselt man temporär mit diesem Befehl die Kennung (substitute user-id). Es wird praktisch ein neuer Login durchgeführt (unsichtbare Eingabe des Kennworts), ohne daß der Begrüßungsbildschirm erscheint. Ohne Angabe einer Kennung wird ein Login für "root" durchgeführt.

Vorsicht: **Nicht die gesamte Arbeitsumgebung wird der neuen Kennung angepaßt.** Beispielsweise werden die Pfade des Stamm-DVZ und des Postkorbs nicht verändert.

Beispiele und Erläuterungen

Eingabe: su gast

(Im folgenden wird das Kennwort abgefragt, dann wird das in der Datei "/etc/passwd" als Initialisierungsprozeß für die betreffende Kennung vermerkte Kommando durchgeführt.)

---

tty	Ausgabe der Nummer des aktuellen Terminals	tty

Es wird der absolute Pfadname der dem aktuellen Prozeß zugeordneten Gerätedatei zur Terminalsteuerung ausgegeben (terminal type). Über diese Datei laufen sämtliche Terminal-Input/Output-Operationen.

Beispiele und Erläuterungen

Eingabe: tty

Ausgabe: /dev/tty00

Hier arbeitet der Benutzer an der Konsole (Nummer "00"). Möglich sind auch Ausgaben wie z. B. "/dev/tty12" oder "/dev/ttyp4". ("p" kennzeichnet bei einem "remote-login" ein virtuelles Terminal.)

---

universe

universe **Aktuelles Universum ausgeben**

Es gibt drei Universen: "sie" (Siemens), "att" (AT&T) und "ucb" (Berkeley). Die Ablaufumgebung der aktuellen Shell muß einem dieser Universen zugeordnet sein.

Beispiele und Erläuterungen

Eingabe: universe

Ausgabe: att

Es wird entweder "sie", "att" oder "ucb" ausgegeben.

---

who

who **Gegenwärtig im System aktive Benutzer anzeigen**

Das "who"-Kommando zeigt Ihnen, wer momentan im System aktiv ist, an welchen Terminals die Benutzer arbeiten und wann die Kennungen den Login durchgeführt haben.

Beispiele und Erläuterungen

Eingabe: who

Ausgabe:

```
maurer      tty06   Jul 30 08:34
lorenz      tty00   Jul 30 08:13
dischinger  tty03   Jul 30 09:11
```

In diesem Beispiel erkennt man z. B., daß der Benutzer "maurer" an der Konsole arbeitet und seit 8:34 Uhr angemeldet ist.

3.2 Bearbeitung des Dateisystems

cat Dateil (...)	Ausgabe von Dateien	cat

Dieser Befehl wird eigentlich nicht zum Lesen benutzt, weil die Ausgabe ohne anzuhalten "durchrauscht" (besser ist die seitenweise Anzeige mit "pg" bzw. "sie more"), sondern zum Zusammenfügen von ASCII-Dateien (concatenate):

cat Dateil (...) >(>) Datei	Dateien verbinden	cat >(>)

Mit diesem Befehl können eine oder mehrere Dateien zusammengefaßt und in eine neue oder schon existierende Datei umgeleitet werden. Wenn die Datei schon vorhanden ist, wird sie mit ">" überschrieben, während mit ">>" die Ausgabe an den bestehenden Inhalt angehängt wird. Existiert die Datei noch nicht, wird sie neu angelegt.

Beispiele und Erläuterungen

"Durchlaufende" Ausgabe der Dateien "kopf" und "fuss":

cat kopf fuss

Zusammenfassen der Dateien "teil1", "teil2", "teil3" zum Zieldokument "Gesamt" (soll überschrieben werden, wenn schon vorhanden):

cat teil1 teil2 teil3 > Gesamt

---

cd (Dateiverzeichnis)	Wechseln des aktuellen DVZ	cd

Mit diesem Kommando können Sie innerhalb des Dateisystems "wandern" (change directory). Wenn kein Dateiverzeichnis angegeben ist, so wechselt Sinix in das Stammverzeichnis.

Beispiele und Erläuterungen

Wechseln vom DVZ "/usr/gast" nach "/usr/gast/ordner":

cd ordner
(oder gleichbedeutend:)
cd /usr/gast/ordner

Wechseln in das übergeordnete Dateiverzeichnis:

cd ..

Wechseln ins Dateiverzeichnis "/usr/att/usr/lib/hit":

cd /usr/att/usr/lib/hit

---

ced

ced (Schalter) (Datei) **Aufruf des Editors CED**

Editor

Ein Editor ist ein Programm zum Erstellen und Ändern von Texten. Mit dem bildschirmorientierten Texteditor CED (C-EDitor") werden Dateien ohne besondere Steuerzeichen (sogenannte "ASCII-Dateien") editiert. Der CED ist in Sinix der Standard-Editor. Beachten Sie, daß z. B. HIT-Dateien damit nicht bearbeitet werden können.

"bildschirm-orientiert"

Man nennt einen Editor "bildschirmorientiert", wenn man den Text wie durch ein verschiebbares Fenster betrachten und mit Hilfe von Cursortasten direkt bearbeiten kann.

Beim ersten Aufruf des CED muß immer ein Dateiname angegeben werden. Danach behält der CED - auch nach dem Logout - den Namen der zuletzt aufgerufenen Datei; die Namen werden bezüglich der Benutzerkennung und des Terminals (!) gemerkt.

Eine Auswahl der wichtigsten Schalter:

-s Suchzeichenfolge: Der Editor sucht die angegebene Zeichenfolge und positioniert den Cursor auf das erste gefundene Muster;

-f Zeilennummer: Der Cursor wird auf die angegebene Zeilennummer positioniert.

Es gibt zwei alternative Möglichkeiten zum Verlassen/Beenden des CED: CED verlassen
Drücken Sie im Texteingabe-Modus auf die <ENDE>-Taste oder drücken Sie die Taste <MENU> und dann "v" für "verlassen" (deutsche Oberfläche) oder "q" für "quit" (englische Oberfläche). In beiden Fällen müssen Sie dann mit "j"/"n" (ja/nein) bzw. "y"/"n" (yes/no) bestätigen, je nachdem, ob Sie sichern wollen oder nicht. - Weitere Hinweise zur Bedienung des CED finden Sie im Handbuch.

<MENU> v
<MENU> q

Beispiele und Erläuterungen

Editieren einer Datei namens "dokument" (bzw. Editieren und Anlegen, soweit noch nicht vorhanden):

ced dokument

Aufruf mit dem Namen der zuletzt bearbeiteten Datei:

ced

chmod

chmod (Wer)[+-=]Was Pfad (...) chmod Oktalzahl Pfad (...)	**Zugriffsberechtigungen modifizieren**

Zugriffsberechtigungen legen fest, wer (welcher Benutzer) was mit einer Datei oder einem Dateiverzeichnis machen kann (Lesen, Schreiben/Löschen oder Ausführen). Voraussetzung zum Ändern der Zugriffsberechtigungen (change mode) ist, daß man entweder Eigentümer der betreffenden Einträge ist oder unter einer der Kennungen "root" und "admin" arbeitet. Sie können die Zugriffseinstellungen mit dem Befehl "ls -l" auflisten lassen und auf zwei Arten ändern:

Symbol-Methode:

Zeichen:
ugoa+-=rwx

"Wer": Mögliche Angaben sind:

u für User (Eigentümer)
g für Group (Gruppe)
o für others (weder Gruppe noch Eigentümer)
a für alle

Der "User" ist der Eigentümer des Eintrags. Er gehört zusammen mit anderen Kennungen einer bestimmten Gruppe an ("group"), die der Systemverwalter festlegt. Kennungen, die dieser Gruppe nicht angehören, werden unter "others" (andere) zusammengefaßt. Wird "Wer" nicht angegeben, so gilt "Was" für alle.

[+-=]: Es steht entweder "+", "-" oder "="

+ : Zugriffsrechte (zusätzlich) vergeben

- : Zugriffsrechte entziehen

= : Zugriffsrechte festsetzen (Ersteinstellung löschen)

"Was": Mögliche Angaben:

r für read (Leseberechtigung)
w für write (Schreibberechtigung)
x für execute (Ausführberechtigung)
u die Rechte des Eigentümers vergeben (kopieren)
g die Rechte der Gruppe vergeben
o die Rechte der anderen vergeben

Oktalzahlen

Oktalzahl-Methode:
Hier sind die den drei Subjektarten zugeordneten Zahlen spaltenweise zu addieren (vgl. Beispiele):

400 = Leseerlaubnis für Eigentümer
200 = Schreiberlaubnis für Eigentümer
100 = Ausführerlaubnis für Eigentümer

040 = Leseerlaubnis für Gruppe
020 = Schreiberlaubnis für Gruppe
010 = Ausführerlaubnis für Gruppe

004 = Leseerlaubnis für andere
002 = Schreiberlaubnis für andere
001 = Ausführerlaubnis für andere

Schematische Darstellung:

Beispiel: "rwx r-- ---" entspricht oktal "740"

(Der Eigentümer hat also vollen Zugriff, die Gruppe nur Lesezugriff, andere haben keine Rechte.)

user	group	others	Wer
r w x 4 2 1	r w x 4 2 1	r w x 4 2 1	Was (symbolisch) Was (oktal)
7 4 +2 +1	4 4 +0 +0	0 0 +0 +0	Oktalzahl Berechnung

Interpretation von Zugriffsrechten:

Zugriffsrechte

Dateien:

r = Lesen (z. B. mit dem Befehl "pg" oder im CED)

w = Schreiben (verändern, löschen; sinnlos ohne "r")

x = Ausführen (nur für Programme relevant)

Dateiverzeichnisse:

r = Lesen (Auflisten der Einträge im Dateiverzeichnis)

w = Schreiben (Anlegen und Löschen eigener Einträge)

x = Ausführen (innerhalb des DVZ Programme ausführen)

Beispiele und Erläuterungen

Wie im folgenden Beispiel deutlich wird, muß man mit der Symbol-Methode ggf. den Befehl mehrfach absetzen, während die Oktalzahl-Methode dies nicht verlangt:
Die Dateien "wichtig" und "../geheim" sollen mit allen Zugriffsberechtigungen (Lesen, Schreiben, Ausführen) nur für den Eigentümer versehen werden:

```
chmod u=rwx wichtig ../geheim
chmod go-rwx wichtig ../geheim
```

(entsprechen:)

```
chmod 700 wichtig ../geheim
```

Die "7" kommt zustande, indem man die Zahlen 4 (Leseerlaubnis für den Eigentümer), 2 (Schreiberlaubnis für den Eigentümer) und 1 (Ausführerlaubnis für den Eigentümer) addiert. Die beiden Nullen bedeuten, daß Gruppenmitglieder und andere keinerlei Rechte erhalten sollen.

Allen Benutzern (also sowohl dem Eigentümer als auch den Mitgliedern seiner Gruppe und anderen Kennungen) wird die Ausführberechtigung für die Prozedur "fueralle" gegeben:

```
chmod +x fueralle
```

Der Gruppe wird für alle Einträge des aktuellen DVZ (durch den Matchcode "*" anzusprechen) und für die der darunterliegenden Verzeichnisebene ("*/*") Schreibberechtigung vergeben (ohne Leseberechtigung wäre dies sinnlos):

```
chmod g+rw * */*
```

Allen Benutzern werden die Zugriffsrechte des Eigentümers für die Dateien des aktuellen DVZ vergeben:

```
chmod a=u *
```

Beispiele für Oktalzahlen:

```
 u   g   o
┌─┐ ┌─┐ ┌─┐
rwx rwx rwx = 777 (u: 4+2+1, g: 4+2+1, o: 4+2+1)
rwx r-- r-- = 744 (u: 4+2+1, g: 4+0+0, o: 4+0+0)
rw- r-x --- = 650 (u: 4+2+0, g: 4+0+1, o: 0+0+0)
r-x --x --x = 511 (u: 4+0+1, g: 0+0+1, o: 0+0+1)
```

sie

copy

copy (Schalter) Quelle (...) Ziel	**DVZ kopieren**

Kopieren der Inhalte eines oder mehrerer Dateiverzeichnisse in ein anderes Dateiverzeichnis. Als Datum der letzten Änderung wird bei den so neu erzeugten Daten das aktuelle gesetzt, die Zugriffsrechte des Originals werden übernommen.

Als "Ziel" bzw. "Quelle" können Dateien oder Dateiverzeichnisse angegeben werden. "copy" entspricht "cp" (Kopieren von Dateien), wenn die Quelle kein DVZ ist.

Möglicher Schalter:

-r: Jedes untergeordnete Dateiverzeichnis wird mit kopiert

Beispiele und Erläuterungen

Das gesamte Dateiverzeichnis "/usr/gast/Dokumente" wird unter dem Namen "Doku" in das aktuelle Dateiverzeichnis kopiert (rekursives Kopieren: mit allen Unterdateiverzeichnissen):

DVZ rekursiv kopieren

sie copy -r /usr/gast/Dokumente Doku

Der Befehl existiert nicht im "att"-Universum! Wenn man dort rekursiv kopieren möchte (z. B. wichtig für INFORMIX-Datenbanken), muß man auf den "cpio"-Befehl in Verbindung mit dem "find"-Befehl ausweichen (s. S. 47).

---

cp Datei1 Datei2 — **Dateien kopieren** — cp Dat/Dat

Die Datei "Datei1" (Quelle) zu "Datei2" (Ziel) kopieren. Als Datum der letzten Änderung wird bei der neu erzeugten Datei das aktuelle gesetzt, die Zugriffsrechte des Originals werden übernommen (copy).

cp Datei1 (...) DVZ — Datei(en) in ein DVZ kopieren — cp Dat/DVZ

Beispiele und Erläuterungen

Kopieren der Datei "befehl" in die Datei "befehl.sav":

cp befehl befehl.sav

Kopieren aller Dateien des aktuellen DVZ nach "/usr/gast":

cp * /usr/gast

---

att

cpio -o(v) — **Dateien/Dateiverzeichnisse sichern** — cpio: copy out

Mit "copy out" wird aus mehreren Dateien bzw. Dateiverzeichnissen eine einzelne Sicherungsdatei erzeugt (die

Quelldaten bleiben erhalten). Die zu sichernden Pfade werden dem Kommando mittels Pipeline (z. B. "ls ¦ cpio -o") oder aus einer Datei (z. B. "cpio -o < Eingabedatei") übergeben. Das Verarbeitungsergebnis (die archivierten Daten) wird standardmäßig am Bildschirm ausgegeben und muß umgeleitet werden (z. B. "ls ¦ cpio -o > Sicherungsdatei").

Schalter:

v : Protokollierung der gesicherten Pfade

att

copy in

`cpio -i(duv(f Muster))` **Dateien/DVZ wieder einlagern**

Mit "copy in" können Daten (Dateien und Dateiverzeichnisse) wieder zurückgespeichert werden. Die Eingabedatei muß mit "copy out" (vorheriger Befehl) erstellt worden sein. Sie wird mittels Pipeline (z. B. "cat Sicherungsdatei ¦ cpio -i") oder mit Hilfe der Eingabe-Umlenkung (z. B. "cpio -i < Sicherungsdatei") an "copy in" übergeben und wieder zu den vorher gesicherten Pfaden expandiert (bleibt aber erhalten).

Schalter und Argumente:

d : Notwendige Dateiverzeichnisse automatisch erzeugen
u : Schon existierende neuere Einträge überschreiben
v : Protokollierung der gesicherten Pfade
f : Nur Einträge, deren Namen zu "Muster" passen, wieder zurückspeichern (z. B. "ordner/doku*")
t : Inhaltsverzeichnis der archivierten Daten ausgeben

att

copy paths

`cpio -p(duv) Ziel-DVZ` **Kopieren von Dateien/DVZ**

Dateien/Dateiverzeichnisse, deren Pfade dem Kommando über die Standard-Eingabe zu übergeben sind (also per Pipeline

oder mit "< Datei"), werden in ein Ziel-Dateiverzeichnis kopiert. Das Ziel-Dateiverzeichnis muß schon existieren. Bei erfolgreicher Ausführung wird die Anzahl der bearbeiteten Blöcke (à 512 Bytes) rückgemeldet.

Mögliche Schalter des Befehls:

d : Notwendige Dateiverzeichnisse automatisch erzeugen

u : Schon existierende neuere Einträge überschreiben

v : Protokollierung der gesicherten Pfade

Beispiele und Erläuterungen

"cpio -pd" bietet sich als Alternative zum "copy -r"-Befehl des Siemens-Universums an:
Kopieren des aktuellen Dateiverzeichnisses mit allen Unterdateiverzeichnissen nach "/usr1/gast/ablage" (das Dateiverzeichnis "ablage" muß schon existieren!):

DVZ rekursiv kopieren

```
find . -print ¦ cpio -pdu /usr1/gast/ablage
```

Der Befehl "cpio" wird oft mit dem "find"-Kommando verbunden, da dieses die benötigten Sicherungspfade in der korrekten Form bereitstellt (pro Zeile ein Pfadname):

```
./ordner
./ordner/datei
usw.
```

Im Beispiel werden alle Pfadnamen unterhalb des aktuellen Dateiverzeichnisses (".") durch das "find"-Kommando erzeugt. Der Schalter "-print" gibt die Pfadnamen aus. Durch die Pipeline ("¦") wird diese Ausgabe an den "cpio -p"-Befehl weitergegeben, so daß alle diese Pfade zum Zielpfad "/usr1/gast/ablage" kopiert werden.

Das Kommando kann auch zum Sichern auf Magnetbandkassetten (MBK) verwendet werden. In diesem Fall ist anstelle der Ausgabedatei die Gerätedatei für das Magnetbandkassettenlaufwerk anzugeben. Sicherung des Inhalts des aktuellen Dateiverzeichnisses (**ohne Unterdateiverzeichnisse**, Dateiverzeichnisse werden überhaupt nicht gesichert):

Sicherung auf eine MBK

```
ls ¦ cpio -o > /dev/rts0
```

Wiedereinspielen dieser Daten mit dem Befehl:

Zurückspeichern von einer MBK

```
cpio -idu < /dev/rts0
```

Inhaltsverzeichnis der gesicherten Daten ausgeben (mit dem Schalter "v" werden hier weitere Informationen ausgegeben, z. B. das Erstellungsdatum der Einträge):

Inhaltsverzeichnis einer MBK

```
cpio -itv < /dev/rts0
```

DVZ archivieren und komprimieren

Eine weitere vorteilhafte Anwendung des Befehls ergibt sich in Verbindung mit dem "pack"-Kommando. Mit den folgenden beiden Kommandozeilen wird zunächst der Inhalt des Dateiverzeichnisses "Dokumente" rekursiv in die Datei "Doku.sic" gesichert und in der folgenden Zeile komprimiert:

```
find Dokumente -print ¦ cpio -o > Doku.sic
pack Doku.sic
```

Wenn man Platz sparen möchte und das Quelldateiverzeichnis in absehbarer Zeit nicht benötigt wird, kann man mit dem "rm -r"-Befehl das Archiv "Dokumente" löschen. Es kann dann jederzeit wieder mit den folgenden Befehlen rekonstruiert werden:

```
unpack Doku.sic.z
cpio -idu < Doku.sic
```

---

sie

crypt

crypt (Schlüssel) < Eingabedatei > Ausgabedatei

Ver- und Entschlüsselung von Dateien

Wenn kein Schlüssel (eine beliebige Zeichenkette als Verschlüsselungswort) angegeben wird, fragt "crypt" danach (unsichtbare Eingabe).

VORSICHT: Wird der Schlüssel vergessen, ist er nicht mehr zu rekonstruieren, und die Datei kann nicht wieder entschlüsselt werden. Wenn eine Datei mehrmals verschlüsselt wird, muß sie in umgekehrter Reihenfolge wieder entschlüsselt werden.
Der Klartext bleibt in der Eingabedatei stehen, während der verschlüsselte Text in die Ausgabedatei geschrieben wird. Wenn die Ursprungsdatei nicht mehr gelesen werden soll, muß sie mit "rm" gelöscht werden.

Beispiele und Erläuterungen

Die Datei "wichtig" soll mit dem Schlüssel "xsichery" verschlüsselt werden; die verschlüsselte Datei wird in der neuen Datei "verschl" abgelegt.

```
sie crypt xsichery < wichtig > verschl
```

Wenn man "xsichery" nicht angibt, fragt das System nach einem unsichtbar einzugebenden Schlüsselwort. Man **entschlüsselt** die Datei "verschl" wieder mit dem folgenden Befehl (das Ergebnis steht dann in der Datei "entschl"):

```
sie crypt xsichery < verschl > entschl
```

---

att

destroy (-i) Datei (...)	**Dateien physikalisch löschen**	destroy

Wenn man Dateien mit "rm" löscht, stehen die Daten im Grunde noch auf der Festplatte und sind in der Dateiverwaltung lediglich als gelöscht markiert. Aus Gründen der Datensicherheit kann es angebracht sein, diese Daten unwiederbringlich mit "Zufallszeichen" zu überschreiben.

Schalter:

-i : interaktives Löschen (für jede Datei wird gefragt)

Beispiele und Erläuterungen

Physikalisches Löschen der Dateien "klausur" und "Protokoll":

```
destroy klausur Protokoll
```

---

ed (Datei)	**Interaktiver, zeilenorientierter Editor**	ed

Ein Editor ist ein Programm zum Erstellen und Ändern von Texten. Im Gegensatz zu bildschirmorientierten Editoren (in denen Cursortasten innerhalb eines "Fensters" benutzt werden können, z. B. CED) muß bei dieser Art von Text-Editoren mit zeilenorientierten Befehlen gearbeitet werden. Aus Platzgründen sollen im folgenden nur wenige Grundfunktionen erläutert werden, die prinzipiell auch für den Stream-Editor "sed" gelten.

Zeilenorientierter Editor

ed-Befehle

Wenn der ed mit einer Datei aufgerufen wird, meldet er die Anzahl der darin enthaltenen Zeichen; wenn die Datei noch nicht existiert, wird sie neu angelegt. Folgende **Befehle** können Sie nach dem Aufruf eingeben (die unterstrichenen Teile sind unveränderlich), wobei ohne Zeilenangabe die aktuelle angenommen wird:

append	(Zeile)a	Hinter der angegebenen Zeile kann ein Text angefügt werden.
change	(Zeilenbereich)c	Der angegebene Zeilenbereich wird durch den folgend einzugebenden Text ersetzt.
delete	(Zeilenbereich)d	Der angegebene Zeilenbereich wird gelöscht.
insert	(Zeile)i	Ein nachfolgend eingegebener Text wird vor der angegebenen Zeilennumer eingegeben.
print	(Zeilenbereich)p	Der angegebene Zeilenbereich wird am Bildschirm ausgegeben.
quit	q = Verlassen des ed mit Rückfrage, falls noch nicht gespeichert wurde (entspricht der Taste <ENDE>).	
Quit	Q = Verlassen des ed ohne Rückfrage (unbedingt).	
read	r Datei = Einlesen einer zu bearbeitenden Datei.	
substitute	(Zeilenbereich)s/Muster/(Zeichenfolge)/(g)	Ersetzen von Mustern mit einer Zeichenfolge innerhalb eines spezifizierten Bereichs. Wenn "g" nicht angegeben wird, wird in jeder Zeile nur das erste Muster ersetzt.
undo	u = Rückgängigmachen des letzten Befehls.	
write	w (Datei) = Schreiben (Sichern) der aktuellen Datei unter dem angegebenen oder (falls "Datei" nicht angegeben wird) dem aktuellen Dateinamen.	

a, i, c beenden

Die **Editierbefehle a, i und c** werden beendet, indem man einen Punkt und <RETURN> eingibt (als erstes und einziges Zeichen in der Zeile) oder die Taste <ENDE> drückt.

Zeilenangaben

Einzelne Zeilen werden durch ihre Zeilennummer spezifiziert (z. B. "5" für Zeile 5, "$" für die letzte Zeile). **Zeilenbereiche** werden angegeben, indem man zwei Zeilennummmern, durch ein Komma getrennt, vor den betreffenden

Befehl setzt (z. B. "3,100" für Zeile 3 bis 100). Man kann für die letzte Zeile symbolisch ein "$" verwenden (z. B. "1,$" für die gesamte Datei, also alle Zeilen).

"$" für letzte Zeile

Beispiele und Erläuterungen

Eine kurze Beispielsitzung soll die Befehlsauflistung verdeutlichen:

Beispielsitzung

Bildschirmausgabe:	Erläuterungen:
ed Testdat	Aufruf aus der Shell
?Testdat	(Systeminformation) Testdat existiert noch nicht
a	Text eingeben
Dies ist Zeile 1. Dies ist Zeile 2. Dies ist Zeile 3. Dies ist Zeile 4. Dies ist Zeile 5.	(Eingegebener Text)
.	Texteingabe beenden
1,$p	Datei ganz anzeigen
Dies ist Zeile 1. Dies ist Zeile 2. Dies ist Zeile 3. Dies ist Zeile 4. Dies ist Zeile 5.	(Angezeigte Zeilen)
1,$s/Dies/Hier/g	In der gesamten Datei das Muster "Dies" durch "Hier" ersetzen
1,$p	Datei (Ergebnis) anzeigen
Hier ist Zeile 1. Hier ist Zeile 2. Hier ist Zeile 3. Hier ist Zeile 4. Hier ist Zeile 5.	(Angezeigte Zeilen)
2,4c	Die Zeilen 2 bis 4 durch die folgenden einzugebenden Zeilen ersetzen
Diese Zeile ersetzt 2 bis 4.	Eingabe der "Ersatzzeile"
.	Ende der Texteingabe
1,$p	Anzeige des Ergebnisses
Hier ist Zeile 1. Diese Zeile ersetzt 2 bis 4. Hier ist Zeile 5.	(Angezeigte Zeilen)
w	Datei mit dem aufgerufenen Namen ("Testdat") sichern
65	(Systeminformation) Anzahl der Zeichen der Datei

fgrep

fgrep (Schalter) Muster Datei1 (...)	Suche nach Mustern

Ohne Schalter werden alle Zeilen, die das Muster (eine beliebige Zeichenkette, z. B. "scr") enthalten, zusammen mit dem Dateinamen ausgegeben (fast global regular expression print). Falls die Musterzeichenkette Leerzeichen enthält, ist sie in Apostrophe zu setzen (" oder ').

Eine Auswahl der Schalter:

-v: Ausgeben aller Zeilen, die das Muster nicht enthalten
-c: Anzahl der Zeilen ausgeben, die das Muster enthalten
-l: Namen der Dateien ausgeben, die das Muster enthalten
-n: Zeilennummer vor jeder Zeile ausgeben

Beispiele und Erläuterungen

Aus den Dateien des aktuellen Dateiverzeichnisses, die mit dem Suffix ".scr" enden, sollen alle Zeilen ausgegeben werden, die die Zeichenfolge "nummer 1" enthalten (zusätzlich sollen Dateiname und Zeilennummer angegeben werden):

```
fgrep -cln "nummer 1" *.scr
```

find

find DVZ1 (...) (Bedingung) Aktion	DVZ durchsuchen

Dateiverzeichnisse nach Namen von Verzeichniseinträgen (Dateien und Dateiverzeichnisse) durchsuchen.

Bedingung:

-name Datei/DVZ: nach dem Namen eines Eintrags suchen
-user Kennung : Dateien suchen, die einem bestimmten Benutzer gehören
-group Gruppe : Dateien suchen, die einer Gruppe gehören
-atime n : Letzter Dateizugriff war vor n Tagen (+n: mehr als n Tage, -n: weniger als n Tage)

Aktion:

-exec : Kommando ausführen, wenn Bedingung erfüllt ist.

Syntax: **-exec Kommando \;**

Die Zeichenfolge " \;" (Leerzeichen, Gegenschrägstrich und Semikolon) schließt das Kommando, das auch eine Kommandoliste oder eine Prozedur sein kann, ab. Im Kommando kann der Name des gefundenen Eintrags mit "{}" (geschwungene Klammern) angesprochen werden.

-print: Den Pfadnamen jedes gefundenen Eintrags ausgeben.

Beispiele und Erläuterungen

Alle Einträge des aktuellen Dateiverzeichnisses mit allen seinen Unterdateiverzeichnissen listen:

```
find . -print
```

Im gesamten Dateisystem alle Einträge finden, die der Kennung "gast" gehören (Vorsicht: nicht überall bestehen Lesezugriffe, deshalb werden entsprechende Fehlermeldungen ausgegeben):

```
find / -user gast -print
```

Alle Einträge in den Dateiverzeichnissen "/usr/gast" und "/usr1/daten", die seit mehr als 20 Tagen nicht verändert wurden, sollen gelöscht werden:

```
find /usr/gast /usr1/daten -atime +20 -exec rm -r {} \;
```

Hier wurde als Kommandoliste nur ein Kommando ("rm -r {}") angegeben. Wenn man z. B. die Namen der gelöschten Einträge am Bildschirm anzeigen möchte, könnte man den Befehl wie folgt modifizieren (mit dem "\" in der ersten Zeile kann man den Befehl über zwei Zeilen schreiben):

```
find /usr/gast /usr1/daten -atime +20 \
-exec echo "Ich lösche jetzt den Eintrag " {}; rm -r {} \;
```

Der Systemverwalter kann mit Hilfe des "find"-Kommandos bequem die durch Kommandofehler entstehenden, in der Regel sehr großen Dateien namens "core" im gesamten Dateisystem löschen. Da das Suchen sehr lange dauern kann, wird der Prozeß mit "&" in den Hintergrund gelegt, so daß man während der Laufzeit des Kommandos weiterarbeiten kann:

core-Dateien löschen

```
find / -name core -exec rm {} \; &
```

lpr (Benutzer)

lpr (Schalter) Datei1 (...) — **Dateien ausdrucken**

Ausgabe von Dateien auf dem Drucker (line printer). Mit diesem Befehl können Sie keine HIT- oder SIPLAN-Dateien ausdrucken, da diese Dateien Steuerzeichen enthalten, die "lpr" nicht interpretieren kann.

Spoolsystem

Spooler

Warteschlange

Der "lpr"-Befehl stellt praktisch das "Spoolsystem" dar. Der "Spooler" ist ein Programm zur Verwaltung von Druckaufträgen, die gesammelt und der Reihe nach ausgegeben werden. Sobald Sie ein Dokument mit "lpr" an den Spooler überweisen, legt dieser eine eindeutige Auftragsidentifikation an, die in einer Warteschlange (englisch "queue") zusammen mit anderen Aufträgen steht. Diese Aufträge werden der Reihe nach an den jeweils zuständigen Drucker geleitet.

Die wichtigsten Schalter (die Reihenfolge ist relevant):

-dru=Druckergruppe	: Druckausgabe auf einer Druckergruppe
-ws=Druckergruppe	: Entspricht Schalter "-dru=..."
-ca Auftragsname	: Löschen des eigenen Druckauftrags
-nc=Zahl	: Anzahl der auszudruckenden Exemplare
-rm	: Löschen der Datei nach dem Ausdrucken
-q	: Informationen über Auftragslage
-qdru	: Informationen über Druckergruppen
-dt oder -int	: deutscher oder internat. Zeichensatz
-pb1, -pb2, -pb3	: Schriftbreite (1 normal, 2 eng, 3 sehr eng)
-tl=Name	: Auftragsname festlegen

lpr

(SV)

"lpr" für den Systemverwalter

Der Systemadministrator kann mit dem "lpr"-Befehl die Druckerverwaltung vollständig kontrollieren. Hierzu gibt es bestimmte für ihn reservierte **Schalter:**

-dg	: Spoolsystem (Druckerbetrieb) beenden
-dk=Drucker:	Drucker sperren mit Abbruch des aktuellen Druckauftrags (wird nach der Freigabe des Druckers zu Ende geführt)
-du=Drucker:	gesperrten Drucker wieder freigeben

-su=Kennung: Benutzerkennung, deren Druckaufträge mit dem Schalter "-ca" gelöscht werden sollen

Anmerkungen: Der **Druckername** ist identisch mit seinem Steckplatz an der Geräterückwand (z. B. "D02" für Steckplatz Nummer "02"). Die Vergabe der Steckplätze kann man im Systemverwalter-Menüsystem mit dem Menüpunkt "aktuelle Konfigurierung abrufen" unter "Konfigurierung des lokalen Systems" erfahren.

Druckername

Das System richtet Drucker**gruppen** automatisch ein. Wenn man z. B. am Steckplatz "02" einen Drucker konfiguriert, wird die Gruppe "G02" mit nur diesem Drucker eingerichtet. Weitere Gruppen müssen im Menü konfiguriert werden.

Druckergruppen

Erklärung der Ausgabe von "lpr -q" anhand eines Beispiels:

lpr -q

```
          D R U C K E R Z U S T A E N D E

DRUCKER  ZUSTAND  LIMIT  ID  AUFTRAG     D-GRUPPE  SEITEN

D08      POLL       2
D09      BEREIT     2

               A U F T R A G S L A G E

ID   AUFTRAG   BENUTZER D_GRUPPE LAENGE  KOP PRIO ZEIT   %

1612 Uebungen  lucifer  STD          4    1   15 13:31  0
```

Im Teil **"Druckerzustände"** ist der Name des jeweiligen Druckers in der Spalte "DRUCKER" angegeben. In der Spalte "ZUSTAND" können mehrere Angaben stehen:

BEREIT: Der Drucker wartet auf Druckaufträge (ist bereit).
LAEUFT: Ein Druckauftrag läuft gerade.
WARTET, Unbekannt, POLL: Kein Drucken möglich, z. B. ist der Drucker off-line geschaltet, es ist kein Papier mehr da, oder das Papier staut sich.

Zustand

Im Teil **"Auftragslage"** ist der Name des Auftrags in der zweiten Spalte ("AUFTRAG") aufgeführt; ebenfalls aufgeführt sind der Benutzer, die Druckergruppe, die Länge des Auftrags, die Anzahl der gewünschten Kopien ("KOP"), die Priorität, die Startzeit des Auftrags und eine Prozentangabe, die angibt, wieviel des aktuellen Druckauftrags bis jetzt ausgeführt wurde.

Auftragslage

Beispiele und Erläuterungen

lpr
Beispiele

Ausdrucken der Dokumente "Dok1" und "Zeitung" mit Schriftbreite 12 Zeichen pro Zoll und nationalem Zeichensatz:

lpr -pb2 -dt Dok1 Zeitung

Informieren über Druckaufträge und Druckerzustände:

lpr -q

Löschen aller eigenen Druckaufträge auf der Druckergruppe "LASER":

lpr -ws=LASER -ca *

Anmerkung: Mit "\" (Gegenschrägstrich) wird das Sonderzeichen "*" entwertet, das normalerweise vom Interpreter durch die Eintragsnamen des aktuellen Dateiverzeichnisses ersetzt wird. In der entwerteten Form werden damit alle Druckaufträge der aufrufenden Kennung angesprochen.

Beispiele für den Systemverwalter:

lpr (SV)

Löschen des Druckauftrags "Dok1" des Benutzers "mgast" auf der Druckergruppe "NR2":

lpr -su=mgast -ca Dok1 -dru=NR2

Spoolsystem beenden:

lpr -dg

Dieser Befehl kann hilfreich sein, wenn das Spoolsystem unerklärliche Fehler aufweist. Man startet das Spoolsystem neu mit dem folgenden Befehl:

/usr/spool/spooler/startup

---

ls

ls (Schalter) (Eintrag1) (...) **Inhalt eines Dateiverzeichnisses ausgeben**

Wenn keine Verzeichniseinträge als Argumente angegeben werden, wird der Inhalt des aktuellen Dateiverzeichnisses ausgegeben. Bei Angabe eines Dateiverzeichnisses wird dessen Inhalt aufgelistet (list).

Eine Auswahl der wichtigsten Schalter:

-a : Ausgabe aller Einträge (auch ".", ".." und alle Einträge, die mit einem "." als erstem Zeichen beginnen)

-d : Anstelle des Inhaltes eines angegebenen Dateiverzeichnisses wird nur dessen Name ausgegeben.

-l : Ausgabe mit Zugriffsberechtigungen, Dateigröße und Erstellungsdatum bzw. Datum der letzten Änderung

-R : rekursive Auflistung aller Unterdateiverzeichnisse mit ihren Inhalten

-x : mehrspaltige Ausgabe (Standard im Siemens-Universum, nicht aber in Kombination mit dem Schalter "-l")

Auflistung von Zugriffsberechtigungen mit "att ls -l":

```
drwxrwxrwx 1 chris   orga    3622    Okt 18 13:23 tutorial
└───1.──┘  2.  3.    4.         5.   └────6.────┘    7.
```

Vgl. chmod

Erklärung der einzelnen Spalten:

1. Einstellung der zehn Zugriffsschutzzeichen. Das erste Zeichen identifiziert den Eintrag:
   - -: der Eintrag ist eine Datei
   - d: der Eintrag ist ein Dateiverzeichnis
   - c: Gerätedatei mit zeichenweisem Zugriff
   - b: Gerätedatei mit blockweisem Zugriff

   Die Interpretation der restlichen (markierten) Zugriffsrechte wird beim "chmod"-Kommando erklärt.
2. Anzahl der Verweise auf den Eintrag (interner Wert).
3. Eigentümer des Eintrags.
4. Gruppe, der der Eigentümer angehört.
5. Größe der Datei in Bytes.
6. Zeitpunkt der letzten Änderung des Eintrags.
7. Name des Eintrags.

Beispiele und Erläuterungen

Den **gesamten** Inhalt des aktuellen DVZ mit Zugriffsberechtigungen rekursiv (mit allen Unter-DVZ) auflisten:

```
ls -alR
```

Im Siemens-Universum kann der Befehl "ls -l" mit "ll" abgekürzt werden, so daß das obige Kommando auch wie folgt geschrieben werden könnte:

```
sie ll -aR
```

Alle mit "ab" oder "brief" beginnenden Namen auflisten, wobei bei Dateiverzeichnissen nicht deren Inhalt, sondern nur deren Einträge ausgegeben werden sollen:

ls -d ab* brief*

Da der "ls"-Befehl in Dateiverzeichnissen mit sehr vielen Einträgen "durchrauscht" (es paßt nur ein Teil der Ausgabe auf den Bildschirm), wird er oft mit "pg" bzw. "sie more" zur bildschirmweisen Anzeige kombiniert:

Eingabe: ls ¦ pg
Ausgabe: Sowohl im AT&T- als auch im Siemens-Universum werden die Einträge bildschirmweise einspaltig angezeigt. (Wenn im Siemens-Universum "ls" in eine Pipeline geleitet wird, geht die Mehrspaltigkeit verloren.)

Eingabe: ls -x ¦ pg
Ausgabe: Die Einträge werden nun bildschirmweise mehrspaltig angezeigt.

---

mkdir

mkdir Dateiverzeichnis **Erstellen eines DVZ**

Legt ein Dateiverzeichnis an (make directory).

Dabei ist ein noch nicht existierender Name anzugeben.

Beispiele und Erläuterungen

Anlegen eines Dateiverzeichnisses mit Namen "ordner":

mkdir ordner

---

more

sie

more Datei1 (...) **Inhalt von Dateien ausgeben**

Die angegebenen (ASCII-)Datei(en) werden am Bildschirm angezeigt. Im Unterschied zu "cat" hält die Ausgabe jedoch bildschirmweise an und wartet auf Benutzereingaben (z. B. zum Blättern). Dateien, die Steuerzeichen enthalten, also z. B. HIT-Dokumente, werden fehlerhaft angezeigt.

Folgend eine **Auswahl der möglichen Eingaben**, während eine Datei seitenweise angezeigt wird:

Leertaste: eine Bildschirmseite weiterblättern
<RETURN> : eine Zeile weiter
q : Verlassen von "more"
h : Ausgeben eines Hilfetextes mit weiteren möglichen Eingaben

Beispiele und Erläuterungen

Bildschirmausgabe der Dateien "Text.alt" und "Neudatei":

more Text.alt Neudatei

---

mv D1 D2	**Umbenennen von Dateien und DVZ**	mv Eintrag

Die Datei bzw. das DVZ "D1" wird in "D2" umbenannt.

mv Datei1 (...) DVZ	**Dateien umsetzen**	mv DVZ

Eine oder mehrere Dateien werden ins Dateiverzeichnis "DVZ" versetzt (move).

Beispiele und Erläuterungen

Dateiverzeichnis "Briefe" in "Korrespondenz" umbenennen:

mv Briefe Korrespondenz

Alle Einträge des aktuellen Dateiverzeichnisses in das Dateiverzeichnis "/usr/gast/allgemein" versetzen:

mv * /usr/gast/allgemein

---

att

pack Datei1 (...)	**Dateien komprimieren**	pack

Der Befehl reduziert den Platz, den die Dateien auf der Festplatte einnehmen. Dies geschieht jedoch nur, wenn sich

das Komprimieren lohnt, d. h., die Dateien müssen groß genug sein (ca. 9 KB). Das Kommando meldet nach dem "Packen" die Kompression in Prozent zurück.
Die komprimierten Dateien werden mit dem Suffix ".z" versehen, die Ursprungsdokumente werden gelöscht. Der Befehl "unpack" entkomprimiert die Daten wieder. Man kann die komprimierten Dateien nicht mehr korrekt mit "pg" oder "more" lesen.

Beispiele und Erläuterungen

Komprimieren der Dateien "riesig" und "unterlagen":

pack riesig unterlagen

Wenn "riesig" groß genug ist, erscheint die Meldung "pack: riesig: xx % Komprimierung".
Wenn "riesig" zu klein für eine Komprimierung ist, wird "pack: riesig: kein Sichern - Datei unveraendert" gemeldet. (Für "unterlagen" gilt Analoges.)

---

att

pg

pg Datei1 (...) **Inhalt von Dateien ausgeben**

Die angegebene(n) Datei(en) werden am Bildschirm angezeigt. Nach jeder Bildschirmseite wartet der Befehl auf Benutzereingaben. Dateien, die Steuerzeichen enthalten, also z. B. HIT-Dokumente, werden fehlerhaft angezeigt.

Eine **Auswahl der möglichen Eingaben**, während eine Datei seitenweise angezeigt wird:

<RETURN> : eine Bildschirmseite weiterblättern
q : Verlassen von "pg"
h : Ausgeben eines Hilfetextes mit weiteren möglichen Eingaben

Beispiele und Erläuterungen

Bildschirmausgabe der Dateien "Text.alt" und "Neudatei":

pg Text.alt Neudatei

---

pr (Schalter) Datei1 (...)	**Dateien druckaufbereitet ausgeben**	pr

Druckaufbereitung bedeutet hier, daß die Dokumente z. B. in Seiten eingeteilt und mit einer Kopfzeile versehen werden. Die Ausgabe erfolgt standardmäßig auf dem Bildschirm. Ohne Schalter: "pr" teilt die Datei in Seiten zu 72 Zeilen auf und gibt eine Kopfzeile (Seitenüberschrift) mit Datum, Dateiname und Seitenzahl aus.

Eine Auswahl wichtiger Schalter:

-h "...": den Kopfzeilentext (in "...") ändern
-lZahl : Dokument in "Zahl" Zeilen pro Seite einteilen
-t : Kopf- und Fußzeilen unterdrücken
+Zahl : die Datei erst ab Seite "Zahl" ausgeben
-Zahl : die Datei in "Zahl" Spalten ausgeben

Beispiele und Erläuterungen

Vier Exemplare der Datei "../ordner/dat1" zweispaltig mit 30 Zeilen pro Seite auf dem Standarddrucker ausdrucken:

pr -l30 -2 ../ordner/dat1 ¦ lpr -nc=4

---

rm (Schalter) (Eintrag)	**Dateien/DVZ löschen**	rm

Wichtige Schalter:

-i : "rm" (remove) fragt ab, bevor gelöscht wird.

-r : Es werden auch Dateiverzeichnisse gelöscht, die jedoch nicht leer sein müssen. Gleichzeitig werden sämtliche Dateien und Unterdateiverzeichnisse des

rm -r rekursiv

angegebenen DVZ gelöscht (rekursives Löschen, d. h., der Befehl ruft sich für sämtliche Unterdateiverzeichnisse selbst nochmals auf und löscht sie).

Wenn Sie einen eigenen Eintrag (Sie müssen selbst Eigentümer sein) löschen wollen, für den Sie keine Schreibberechtigung besitzen, fragt Sinix vorher ab, ob wirklich gelöscht werden soll. Wenn Sie "y"/"j" für "yes"/"ja" eingeben, wird die Datei gelöscht, andernfalls nicht.

Beispiele und Erläuterungen

Löschen der Dateien "/usr/gast/datei12" und "aerger":

rm /usr/gast/datei12 aerger

Löschen des Dateiverzeichnisses "Tabellen" (rekursiv):

rm -r Tabellen

Besonders als Systemverwalter sollte man vorsichtig sein (z. B. würde "rm -r /" das gesamte Dateisystem löschen)!

Der "rm -r"-Befehl dient auch dazu, die "Platte zu putzen", d. h. überflüssige Benutzerdaten zu entfernen, um Platz für neue Daten zu schaffen. Man kann den Befehl mit Matchcodes kombinieren:

rm -r /usr/inform/* /usr/inform/.[!.]* /usr/inform/..?*

Mit diesen Matchcodes werden sämtliche Verzeichniseinträge erfaßt (auch Einträge, deren Namen mit einem "." beginnen, aber nicht die Einträge "." und ".."), so daß mit dem obigen Befehl sämtliche Einträge unter "/usr/inform" gelöscht werden.

---

sed
Stream-Editor

sed -n (Schalter) Datei Zeilenorientierter Stream-Editor

"sed" wird nicht interaktiv (im Gegensatz zum ed), sondern prozedural eingesetzt, d. h., er eignet sich besonders zur Verwendung in Shell-Prozeduren (**s**tream **ed**itor). Seine Befehle entsprechen weitgehend den Befehlen des "ed", soweit diese nicht interaktiv sind (z. B. "a(ppend)"). Wie beim "ed" sollen hier aus Platzgründen nur die grundsätzlichen Züge der Funktionsweise beschrieben werden.

Schalter:

-n : Die Bildschirmausgabe der eingelesenen Zeilen wird unterdrückt.

-f Befehlsdatei: Der sed liest seine Befehle aus der Datei "Befehlsdatei".

-e 'Befehl' : Der in Hochkommata eingeschlossene Befehl wird ausgeführt (auch " möglich).

Die Schalter "-f" und "-e" können beliebig oft wiederholt werden. Das Bearbeitungsergebnis wird auf der Standard-Ausgabe (dem Bildschirm) ausgegeben.

Beispiele und Erläuterungen

In der Datei "Testdat" (vgl. Kommando "ed") soll die Zeichenfolge "Hier" mit "Sehen Sie nur: " ersetzt werden:

```
sed -e '1,$s/Hier/Sehen Sie nur: /' -e 'w temp' Testdat
mv temp Testdat
```

Da mit dem sed-Befehl "w" (write) die Eingabedatei nicht unter ihrem ursprünglichen Namen gesichert werden kann, muß sie zunächst unter einem anderen Namen abgespeichert und dann mit dem Kommando "mv" zurückbenannt werden.

Die Editoren ed und sed bieten **erheblich** mehr Möglichkeiten, als hier gezeigt werden kann. Beispielsweise kann mit sogenannten regulären Ausdrücken gearbeitet werden, die praktisch Matchcodes für eine Menge an Zeichenfolgen darstellen. Um dies andeutungsweise zu illustrieren, folgt hier ein Beispiel:

Die Datei "brief1" soll in jeder Zeile am Zeilenanfang um 5 Leerzeichen eingerückt werden, und alle Leerzeilen sollen mit 5 Sternen markiert werden. Das Ergebnis der Bearbeitung soll in der Datei "brief2" stehen:

```
sed -e '1,$s/^$/*****/' -e '1,$s/^/     /' brief1 > brief2
    └──────────────────┘ └─────────────────┘        └──────┘
          1. Befehl            2. Befehl            Umleitung
```

1. Befehl: Mit "1,$" werden alle Zeilen der Datei adressiert (erste bis letzte Zeile). "s/.../.../" ist der Ersetzungsbefehl. "^" ist der reguläre Ausdruck für den Zeilenanfang, "$" steht für das Zeilenende (nicht zu verwechseln mit dem "$" bei der Zeilenbereichsspezifikation). Da zwischen Anfang und Ende der Zeile laut diesem regulären Ausdruck keine Zeichen stehen dürfen, werden alle Leerzeilen der Datei erfaßt. Dort werden die 5 Sterne eingetragen.

2. Befehl: Am Anfang jeder Zeile (wieder "^") werden die 5 Leerzeichen eingetragen.
Umleitung: Die Ausgabe wird mit dem Zeichen ">" in die Datei "brief2" umgeleitet. Alternativ zur Umleitung kann man auch "-e 'w brief2'" schreiben.

---

unpack

att

unpack Dateil.z (...)	**Dateien dekomprimieren**

Dateien, die mit dem Kommando "pack" komprimiert wurden, können mit dem Befehl "unpack" wieder entkomprimiert werden. Die mit "pack" bearbeiteten Dateien sind mit dem Suffix ".z" versehen.

Beispiele und Erläuterungen

Entkomprimieren der Dateien "riesig" und "loesungen":

unpack riesig.z loesungen.z

---

vi
Unix-Standard-Editor

att

vi Datei	**Unix-Standard-Editor**

Der "vi" ist in vielen Unix-Systemen als Standard-Editor zu finden. Hier sollen nur grundsätzliche Punkte angesprochen werden. Geben Sie nach dem Aufruf "**a**" ein, um den **Text direkt bearbeiten** zu können. Sie können die Einfüge- und Löschtasten sowie die einfachen Pfeiltasten benutzen.

Der "vi" ist sowohl zeilen- als auch bildschirmorientiert. Wenn Sie die Escape-Taste (**<ESC>**) drücken und direkt danach einen Doppelpunkt (":") eingeben, gelangen Sie in den Kommando-Modus, der die Eingabe von Befehlen erlaubt. Der Befehlsvorrat ist den "ed"-Kommandos sehr ähnlich (z. B. der "substitute"-Befehl "s").

Befehle des Kommando-Modus:

(Aufruf mit <ESC>)

q = vi verlassen, nachdem gesichert wurde

q! = Unbedingtes Beenden (ohne zu sichern)

w (Datei) = Sichern der Datei (ohne "Datei": Aufrufname)

Wenn Sie aus dem Kommando-Modus wieder in den Editier-Modus zurück wollen, drücken Sie <RETURN> und geben Sie nochmals ein "a" ein.

wc (Schalter) Datei (...)	**Wörter/Zeichen/Zeilen zählen**	wc

Ohne Schalter werden drei Ziffern für Zeilen, Wörter und Zeichen ausgegeben (word counter).

Mögliche Schalter:

-l: nur Zeilen zählen (lines)

-w: nur Wörter zählen (words)

-c: nur Zeichen zählen (characters)

Beispiele und Erläuterungen

Die Zeilen und Wörter der Dateien "drdat" und "x12" zählen:

wc -lw drdat x12

3.3 Bearbeitung von Disketten und Magnetbandkassetten

/etc/flformat
/etc/format

/etc/flformat /etc/format	Disketten formatieren

Formatieren heißt, eine Diskette sektorenweise in Boot-, Label- und Datenbereich einzuteilen, so daß sie für das Betriebssystem lesbar und beschreibbar ist (floppy formatter). Der vorherige Inhalt der Diskette wird gelöscht. Das Formatieren dauert mehrere Minuten. Für jeden formatierten Sektor schreibt Sinix einen Punkt auf den Bildschirm. In In Sinix V5.2x heißt der Befehl "/etc/flformat", in Sinix V5.4 "/etc/format".

Folgende Diskettenart kann immer verwendet werden:
- Double sided und double density
- 5 ¼ Zoll Durchmesser
- 48 tpi ("tracks per inch" = Spuren pro Zoll)

sie

far

far [crxt](v) Pfade	Lesen/Schreiben auf Diskette

Den Datenbereich der Diskette bearbeiten (floppy archiver). Wenn die Diskette voll ist, wird automatisch eine neue angefordert. Falls die Diskette nicht alle Daten aufnehmen kann, werden Dateien auf mehrere Disketten aufgeteilt. Zur Erläuterung der Schalter siehe unten.

tar
Diskette

tar [crxt](v)f /dev/fl2 Pfade	Lesen/Schreiben auf Diskette (floppy 2)

Die Syntax entspricht bis auf die Angabe "f /dev/fl2" der des "far"-Befehls. Hier werden jedoch zu große Dateien nicht automatisch auf mehrere Disketten aufgeteilt (tape archiver). Zur Erläuterung der Schalter siehe unten.

V5.2x (V5.40)

tar [cxt](v)f /dev/rts0 Pfade tar [cxt](v)f /dev/tape Pfade	Lesen/Schreiben auf Magnetbandkassette	tar MB-Kassette

Die Syntax entspricht bis auf die Angabe "f /dev/rts0" der des "far-Befehls". Beachten Sie aber, daß hier der Schalter "r" nicht funktioniert.

Erläuterung für "far" und "tar":
Es können immer mehrere Dateien und Dateiverzeichnisse angegeben werden. Die Dateiverzeichnisse werden automatisch rekursiv gesichert (alle unter einem Ordner zu findenden Einträge werden ebenfalls auf die Diskette kopiert). Beachten Sie, daß hier kein "-" vor den Schaltern angegeben wird.

Schalter:
c: Überschreiben des vorherigen Inhalts mit Daten
r: Anhängen neuer Daten an die Daten auf der Diskette
t: Inhaltsverzeichnis des Datenträgers ausgeben
v: erweitertes Protokollieren der Aktionen
x: Einlesen der angegebenen Dateien/DVZ auf die Platte
f /dev/rts0: Angabe der Gerätedatei (MBK-Laufwerk), V5.2x
f /dev/tape: Angabe der Gerätedatei (MBK-Laufwerk), V5.40

far/tar r

Mit dem Schalter "r" (Anhängen an bestehende Daten auf dem Datenträger) kann man Dateien mit gleichen Pfadnamen auch mehrfach auf einer Diskette speichern. Beim Wiedereinspielen der Daten auf die Festplatte macht Sinix jedoch keinen Unterschied zwischen diesen Pfaden und überschreibt mit den zu einem späteren Zeitpunkt gespeicherten Einträgen die zeitlich zuerst gesicherten Daten.

Fehlerbild

Anmerkung: Auch wenn keine Diskette im Laufwerk liegt, versucht das Betriebssystem zu lesen. Auf dem Bildschirm der aufrufenden Kennung erscheint in diesem Fall erst nach einer gewissen Zeit eine Fehlermeldung, selbst wenn mit <DEL> abgebrochen wird; dagegen erscheinen andere Fehler-

meldungen sofort auf der Konsole (auch wenn eine Diskette bzw. eine Magnetbandkassette schadhaft oder unformatiert ist).

Beispiele und Erläuterungen

Die Datei "Tabelle.alt" und das Dateiverzeichnis "Dokumente" sollen auf Diskette gesichert werden, wobei der alte Inhalt zu überschreiben ist und die Aktionen erweitert protokolliert werden sollen (z. B. sollen die Zugriffsrechte der gesicherten Pfade angegeben werden). Der Befehl lautet im Siemens-Universum:

```
sie far cv Tabelle.alt Dokumente
```

Im AT&T-Universum muß der Befehl wie folgt lauten:

```
tar cvf /dev/fl2 Tabelle.alt Dokumente
```

Einlesen der Datei "brief" und der "teile":

```
tar xf /dev/fl2 brief teile
```

Ausgeben eines Inhaltsverzeichnisses der Diskette:

Eingabe: `tar tvf /dev/fl2`

Ausgabe: (z. B.)

```
-rw-------247/15 151523 Aug  7 10:54 1991 shell
-rw-------247/15  16883 Aug  7 09:44 1991 Uebungen
-rw-------247/15  11257 Aug  2 08:23 1991 sinikurz
```

Zugriffe | User-/Group-ID | Größe in Bytes | Zeit und Datum der Sicherung | Name des Eintrags

Die Pfade "/usr1/lucifer" und "/usr1/sicher" sollen auf Magnetbandkassette gesichert werden:

```
tar cvf /dev/rts0 /usr1/lucifer /usr1/sicher
```

Wiedereinlesen der vorher gespeicherten Daten:

```
tar xvf /dev/rts0
```

3.4 Software-Aufrufe in Sinix

Am verbreitetsten in Sinix sind zweifellos die Software-Produkte HIT V4.xx (Textsystem), INFORMIX V2.1 (Datenbanksystem) und SIPLAN V2.0 (Tabellenkalkulation). Deshalb werden im folgenden die wichtigsten diesbezüglichen Kommandos dargestellt.

att

fit	**HIT-Menü aufrufen**	fit HIT-Menü

Ruft das HIT-Menü auf, in dem bestimmte Konfigurationen zum Aufruf des HIT-Texteditors (vgl. Kommando "hit") eingestellt werden können (z. B. aufzurufendes Dokument, Formatiertabelle). Mit diesem Menü erstellte Archive sind Dateiverzeichnisse, die direkt unter dem Stamm-DVZ der aufrufenden Kennung abgelegt werden. Sie beenden das Menü, indem Sie zweimal die Taste <ENDE> betätigen.

att

hit (Datei1) (...)	**HIT-Editor aufrufen**	hit HIT-Editor

Mit diesem Befehl wird der HIT-Editor aufgerufen (Sie verlassen ihn mit <ENDE>), in dem man Texte bearbeiten kann. Wenn mehrere Dateien angegeben werden, fragt HIT nach dem Verlassen des ersten Dokuments nach, ob die nächste Datei bearbeitet werden soll. HIT-Dokumente haben ein eigenes Format, sind also nicht mit "normalen" Shell-Befehlen wie "pg" oder "fgrep" zu bearbeiten.

att

hless
HIT-Dateien
ansehen

hless Datei	**HIT-Datei aus der Shell ansehen**

Die Datei wird bildschirmweise angezeigt. Neben den auch in HIT gültigen Cursortasten sind während des Ansehens folgende Eingaben möglich:

<HILFE> : Ausgabe eines Hilfe-Textes
h : Aufruf des HIT-Editors
<LEERTASTE> : Einen Bildschirm weiter

isql DB

isql SQL-Datei

INFORMIX

isql (Datenbank)	**INFORMIX-Menü aufrufen**
isql Datenbank SQL-Datei	**SQL-Datei ausführen**

Mit der ersten Variante wird das Menü des Datenbanksystems INFORMIX aufgerufen. Die zweite Variante des Kommandos führt eine SQL-Datei aus und gibt die Ausgabe am Bildschirm aus. SQL-Dateien sind (z. B. mit dem Editor CED erstellte) ASCII-Dateien (ohne Steuerzeichen), die Befehle in der Datenbankabfragesprache "SQL" (Structured Query Language) enthalten. Die SQL-Befehle können von der Shell natürlich nicht direkt interpretiert werden, sondern müssen über das "isql"-Kommando zur Ausführung an INFORMIX übergeben werden.
Die Datenbank (in der Shell ein Dateiverzeichnis, dessen Name mit dem Suffix ".dbs" endet und das bestimmte INFORMIX-Systemdateien enthält) muß entweder im aktuellen Dateiverzeichnis oder unter einem in der Shell-Variablen "$DBPATH" definierten Pfad zu finden sein.

<u>Beispiele und Erläuterungen</u>

Aufruf des INFORMIX-Menüs mit der Datenbank "firma":

isql firma

Definition von Datenbankpfaden (Dateiverzeichnisse, unter denen die Datenbanken zu finden sind; damit die Variable auch den folgenden Shell-Prozessen bekannt ist, wird sie mit dem Befehl "export" bekannt gemacht):

```
DBPATH=/usr1/dbsys/DB:/usr1/lucifer/DB.ordner:/usr/daten
export DBPATH
```

Die Definition dieser Variable sollte sinnvollerweise in der Datei "$HOME/.profile" geschehen.

Ausführung der SQL-Datei "anzahl.sql" (für die Datenbank "firma"; das Suffix ".sql" ist obligatorisch) und Umleitung des Ergebnisses in die Datei "anzahl.erg":

```
isql firma anzahl.sql > anzahl.erg
```

---

sacego (-q) Listenprogramm | **INFORMIX-Listenprogramm ausführen**

sacego
INFORMIX-
Listenprogramm

Das Listenprogramm erzeugt einen Listenoutput (formatierte Auswertung der Daten einer Datenbank), der gemäß der im Programm festgelegten Anweisungen ausgegeben wird. Das Listenprogramm muß übersetzt worden sein, und es müssen Ausführzugriffe bestehen.

Schalter:

-q : Unterdrückung der Ausgabe von Bildschirmmeldungen

Beispiele und Erläuterungen

Aufruf des Listenprogramms "liste" (Meldungen von INFORMIX sollen unterdrückt werden):

```
sacego -q liste
```

siplan
SIPLAN-Menü

siplan (Arbeitsblatt-Datei)	SIPLAN aufrufen

Mit diesem Kommando wird das Tabellenkalkulationsprogramm SIPLAN aufgerufen (man verläßt es mit der <ENDE>-Taste). SIPLAN-Arbeitsblätter ("Spreadsheets") sind Dateien, die nur für das Software-Produkt SIPLAN verständlich sind. Sie liegen also nicht im ASCII-Format vor und können z. B. nicht mit dem CED bearbeitet werden. Mit der automatischen Tabellenkalkulation können Berechnungen für Tabellen mit Zahlenwerten (auch in Form komplexer Formeln) auf komfortable Art und Weise durchgeführt werden.

sperform
INFORMIX-Formate

sperform Maskenprogramm	INFORMIX-Maskenprogramm ausführen

Maskenprogramme ("Formatprogramme") zeigen am Bildschirm Formulare ("Formate") an, die mit Hilfe automatischer Arbeitsfunktionen (z. B. für das Suchen von Daten) eine direkte Bearbeitung der Daten einer Datenbank ermöglichen. Das INFORMIX-Format muß als kompiliertes Maskenprogramm vorliegen (mit Ausführberechtigung für den Aufrufenden).

Beispiele und Erläuterungen

Aufruf des Maskenprogramms "maske01":

```
sperform maske01
```

Mit Hilfe von Umgebungsvariablen kann man Funktionen des Maskenprogramms steuern. Zum Beispiel ist die vorherige Definition eines Druckprogramms oft nützlich (hier mit "pr", "lpr"):

```
DBPRINT='pr -l64 -h "Format-Output" | lpr -pb2'
export DBPRINT
```

Wenn man nun im PERFORM-Menü den Punkt "Print" anwählt, wird statt des Standard-Druck-Programms "lpr" die obige Zeile angeboten. Wie bei der Variablen "DBPATH" (vgl. das Beispiel zum "isql"-Befehl) empfiehlt sich die Definition in der Datei "$HOME/.profile".

3.5 Kommunikation in Sinix

Die im folgenden vorgestellten Kommandos stellen nur einen kleinen Ausschnitt der Kommunikation in Sinix-Systemen dar, sind aber die erfahrungsgemäß am häufigsten gebrauchten. Es werden die üblichen Befehle des Betriebssystems erläutert und - da Local Area Networks (LANs) immer weitere Verbreitung finden - auch LAN-Kommandos (basierend auf dem Software-Produkt REMOS/CCP-LAN V3.xx). LAN

Wenn von einem "Host" die Rede ist, so ist im folgenden damit ein ferner Rechner im Sinne eines LAN gemeint (im Unterschied zum lokalen Rechner, auf dem man gerade arbeitet). Host

"r"-Befehle: Datei "$HOME/.rhosts" remote-Befehle

Kommandos, die mit einem "r" beginnen, sind im allgemeinen LAN-Befehle ("r" für remote, also "fern"). Bevor man die Kommandos **rcp**, **rlogin**, **rsh**, **rtar** korrekt rechnerweit nutzen kann, muß für die jeweiligen Netzbenutzer eine Autorisierungsdatei angelegt werden, die die rechnerübergreifenden Zugriffe regelt (s. u.).

Um Netzkommandos auf einem anderen Rechner ausführen zu dürfen, muß man Zugang zu einer Kennung auf dem Host haben, unter deren Zugriffsberechtigung der Befehl dann abläuft. Die Kennung auf dem lokalen Rechner stößt die Ausführung des Kommandos auf dem fernen Rechner nur an, die entsprechenden Prozesse laufen dann ab, als hätte die Kennung auf dem Host den Befehl abgesetzt. Trägerkennung

Wenn Sie anderen Benutzern von fernen Rechnern erlauben wollen, unter Ihrer Kennung auf dem lokalen Rechner Kommandos auszuführen, müssen Sie in der Datei ".rhosts" unter Ihrem Stamm-Dateiverzeichnis die Kennungen dieser Rechner eintragen. $HOME/.rhosts

Syntax der Einträge in "$HOME/.rhosts":

```
Rechnername1 Kennung1
Rechnername2 Kennung2
(...)
```

Sie könnten die Datei z. B. mit dem folgenden Inhalt anlegen:

```
mx3003 admin
mx3005 gast
hamburg peer
```

Im Beispiel können die Kennungen "admin", "gast" und "peer" von den besagten Rechnern aus Befehle absetzen und auch einen Login unter ihrer Kennung durchführen.

!! VORSICHT !!
Im Interesse der Datensicherheit wird empfohlen, derartige Einträge vor allem bei "admin" und "root" genau zu überprüfen, da sie ein unerlaubtes Eindringen in das System begünstigen: Es wird ja kein Kennwort mehr abgefragt.

Matchcodes
in r-Befehlen

Bei der Angabe von Matchcodes für Verzeichniseinträge (vor allem "*", "?") werden Metazeichen, die nicht mit dem Gegenschrägstrich ("\") oder mit Apostrophierungszeichen (' bzw. ") entwertet wurden, auf dem lokalen Rechner interpretiert, entwertete Metazeichen dagegen auf dem fernen Rechner. Beispiele hierzu finden Sie beim "rsh"-Kommando.

ftp
File-Transfer

ftp (Host)

File-Transfer-Programm aufrufen

Ein "File-Transfer" ist ein direkter Datenaustausch zwischen zwei Rechnern. Wenn die Verbindung korrekt aufgebaut wurde (Informationen über erreichbare ferne Rechner können mit dem Kommando "ruptime" abgerufen werden), wird nach einer Kennung im fernen System gefragt. Per Voreinstellung wird der Name der eigenen Kennung angenommen. Sie können mit <RETURN> bestätigen oder eine andere Kennung eingeben. Das Kennwort muß dann unsichtbar eingegeben werden.

Sobald die Verbindung steht (der Prompt "ftp>" wird ausgegeben), kann man Kommandos eingeben. Mit der Eingabe "?" erhalten Sie eine Liste der möglichen Kommandos, bei-

spielsweise "cd", um das Dateiverzeichnis auf dem fernen Rechner zu wechseln, "lcd" für einen Verzeichniswechsel auf dem lokalen Rechner und "get Datei" zum Transferieren einer Datei zum lokalen Rechner.

hostname **Ausgabe des Namens des aktuellen Rechners** hostname

Jeder Rechner in einem LAN besitzt einen eindeutigen Namen, mit dem er angesprochen werden kann. Wenn man den Namen des lokalen Rechners nicht kennt, kann man ihn mit dem Kommando "hostname" abrufen.

Beispiele und Erläuterungen

Eingabe: hostname

Ausgabe: mx3006

mail **Elektronische Post lesen**

Wenn Briefe vorhanden sind, werden sie nacheinander angezeigt. Das Programm ist interaktiv: Wenn der Prompt "?" ausgegeben wird, können Sie unter anderem folgende Eingaben tätigen:

? : Hilfe-Informationen über mögliche Eingaben
+ : nächster Brief (aktueller Brief bleibt im Postkorb)
d : Löschen des aktuellen Briefes
q : (quit) Verlassen der Postfunktion
w Datei : Abspeichern des aktuellen Briefes in Datei (der Brief ist dann nicht mehr im Postkorb vorhanden)

mail

mail Kennung(@Host) (...) < Datei	Post versenden

Der Inhalt von "Datei" wird an die angegebenen Benutzer netzweit oder lokal gesendet. Wenn man eine Datei als "Brief" zu einem anderen Rechner verschicken will, muß man zusätzlich zur Empfänger-Kennung den Zielrechner angeben, auf dem diese Kennung arbeitet (den Host). Vor dem Hostnamen steht das "@" ("At"-Zeichen, "Klammeraffe"). Mit "<" wird die Eingabe aus der darauf folgenden Datei gelesen.

Jeder Brief wird automatisch mit einem Kopfteil versehen, in dem u. a. die Absendezeit, der Sender und der Empfänger vermerkt werden.

Beispiele und Erläuterungen

Absenden der Datei "wichtig" als elektronische Post zu den Kennungen "gast" und "admin" auf dem lokalen Rechner:

mail gast admin < wichtig

Dasselbe an die Kennung "root" auf dem Rechner "vertrieb":

mail root@vertrieb < wichtig

Lesen der eigenen elektronischen Post:

Bildschirm-Ein-/Ausgabe:	**Erläuterungen:**
$ mail	Programmaufruf
From gast Mon Jul 29 17:18:41 1991 Date: Mon, 29 Jul 91 17:18:39 +0100 From: <gast> Apparently-To: gerd	Briefkopf (durch System erzeugt)
Hallo! Dies ist der erste Brief! Ende.	Briefzeilen
? d	Löschen des Briefes
From gast Mon Jul 29 17:18:36 1991 Date: Mon, 29 Jul 91 17:18:33 +0100 From: <gast> Apparently-To: gerd Hallo! Dies ist der zweite Brief! Ende.	Anzeige des zweiten Briefes im Postkorb

? w brief	Sichern des Briefes als Datei mit dem Namen "brief"
$	Verlassen des Programms

mesg ([yn])	**Nachrichtenempfang zulassen/verweigern**	mesg

Mit dem Befehl "mesg" können Sie entscheiden, ob andere Benutzer Ihnen mit dem Befehl "write" Nachrichten auf Ihren Bildschirm senden können (messages). Wenn Sie keinen Schalter angeben, erfahren Sie, ob Sie Nachrichtenempfang zulassen oder verweigern. Der Schalter "y" steht für yes, "n" für no.

rcp Quelldatei Zieldatei	**Datei netzweit kopieren**	rcp Datei
rcp (-r) Quellpfad (...) Ziel-DVZ	**Dateien/DVZ netzweit kopieren**	rcp DVZ

Sie können dieses Kommando nur ausführen, wenn Sie eine Zielkennung auf dem fernen Rechner besitzen, d. h., Sie müssen in der Datei "$HOME/.rhosts" bei der Zielkennung eingetragen sein (s. o.).

Schalter:

-r : Rekursives Kopieren von DVZ (mit allen Unter-DVZ)

Format der Pfadangaben:

Ferner_Rechner.Ferne_Kennung:Pfad

Beispiele und Erläuterungen

Kopieren der lokalen Datei "Briefe/anVorstand" zum fernen Rechner "MX300-1" (ferne Kennung: "gast") in das (existierende) Dateiverzeichnis "/usr1/gast/Verkehr":

rcp Briefe/anVorstand MX300-1.gast:/usr1/gast/Verkehr

Rekursives Kopieren des Dateiverzeichnisses "/usr1/petra/Angebote" vom Rechner "Vertrieb" über die Kennung "petra" zum lokalen Rechner in das (existierende) Dateiverzeichnis "Petra.Ang":

Rekursives Kopieren

rcp -r Vertrieb.petra:/usr1/petra/Angebote Petra.Ang

Rekursives Kopieren aller Dokumente und Dateiverzeichnisse (bis auf die, die mit einem Punkt beginnen) der Kennung "personal" auf dem fernen Rechner "zentrale" in das aktuelle Dateiverzeichnis auf dem lokalen Rechner:

rcp -r zentrale.personal:/usr1/personal/"*" .

rlogin

rlogin Host (-l Kennung) **Am fernen Rechner anmelden**

Mit diesem Kommando können Sie einen vollwertigen Login am fernen Rechner durchführen. Ihre Prozesse auf dem lokalen Rechner bleiben im Hintergrund aktiv, bis Sie die Sitzung am fernen Rechner beendet haben. Durch die Leistungsfähigkeit der verwendeten Software und die eingestellte Übertragungsrate bedingt, können sich die Antwortzeiten verlängern.

Argumente:

Host : Name des fernen Rechners
-l Kennung : Login-Kennung am fernen Rechner

Wenn der aufrufende lokale Benutzer bei der Kennung am fernen Rechner in der Datei "$HOME/.rhosts" eingetragen ist, wird kein Kennwort abgefragt.

Beispiele und Erläuterungen

Login am fernen Rechner "MX3003" unter der Kennung "gast":

rlogin MX3003 -l gast

---

rsh Host (-n) -l Kennung Kommando	Kommando auf einem fernen Rechner ausführen	rsh

Der "rsh"-Befehl führt Kommandos unter der angegebenen Kennung auf dem fernen Rechner aus. Dazu muß die aufrufende Kennung des lokalen Rechners in der Datei "HOME/.rhosts" bei der Kennung auf dem Host eingetragen sein.

Argumente:

Host : ferner Rechner

-n : muß angegeben werden, wenn die Ausgabe des Kommandos rsh über eine Pipe an ein weiteres Kommando weitergegeben werden soll oder wenn das am fernen Rechner ausgeführte Kommando interaktiv funktioniert (vgl. Beispiele)

-l Kennung : Login-Kennung am fernen Rechner

Beispiele und Erläuterungen

Auflisten des Inhalts des Stamm-DVZ der Kennung "gerd" auf dem Host "mx3003" mit dem "ls -l"-Kommando:

rsh mx3003 -l gerd ls -l

Anzeigen aller Dateien im Dateiverzeichnis "/usr2/prof/info" der Kennung "prof" auf dem fernen Rechner "FB03" mit dem Kommando "pg". Da "*" für die Dateien auf dem fernen Rechner steht, muß dieses Sonderzeichen entwertet werden. Hier muß "-n" angegeben werden, da das interaktive Kommando "pg" sonst am fernen Rechner auf Eingaben warten würde; der erste "pg"-Befehl bewirkt auf dem fernen Rechner keine seitenweise Ausgabe, sondern die Anzeige der Dateinamen, während der zweite "pg"-Befehl bildschirmweise anzeigt (cat würde alle Dateien zusammengefaßt anzeigen, ohne dazwischen deren Namen auszugeben):

rsh FB03 -n -l prof pg /usr2/prof/info/* ¦ pg

Metazeichen bei rsh

Die Interpretation von Metazeichen betrifft auch die Umleitung der Standard-Ein- und -Ausgabe. Im folgenden Beispiel wird die Datei "/usr1/texte/briefkopf" auf dem fernen Rechner "R2" an die lokale Datei "anVorstand" angehängt (die ferne Kennung heißt "koenig"):

```
rsh R2 -l koenig cat /usr1/texte/briefkopf >> anVorstand
```

Wenn die Datei "briefkopf" an eine Datei "anVorstand" auf dem fernen Rechner angehängt werden soll, müssen die Metazeichen für den lokalen Rechner entwertet werden:

```
rsh R2 -l koenig cat /usr1/texte/briefkopf ">>" anVorstand
```

"anVorstand" ist hier ein relativer Name, der von Sinix zu "$HOME/anVorstand" im fernen System expandiert wird.

Drucken auf fernen Rechnern

Ein wichtiges Anwendungsgebiet für den "rsh"-Befehl ist das Drucken auf fernen Rechnern:

```
cat Datei ¦ rsh R2 -l gast lpr -dru=G11
```

Mit "cat" wird die auszudruckende Datei über eine Pipeline an den "rsh"-Befehl weitergegeben, der die Daten wiederum dem "lpr"-Kommando des fernen Rechners übergibt.

---

rtar

rtar Host -l Kennung -Schalter (Pfade)

Datensicherung auf einem fernen Rechner

"rtar" erlaubt die Nutzung von Disketten- und Magnetbandkassettenlaufwerken auf fernen Rechnern. Schalter und Pfade werden wie beim "tar"-Kommando angegeben. Auf dem Host wird ein Eintrag in der Datei ".rhosts" unter dem Stamm-Dateiverzeichnis der beim fernen Rechner angesprochenen Kennung benötigt.
Beachten Sie, daß im Unterschied zu "tar" hier die Schalter mit "-" eingeleitet werden.

Argumente:

Host	: ferner Rechner
-l Kennung	: Kennung auf dem fernen Rechner
Schalter	: -[crxt](v)f Gerätedatei
Pfade	: zu sichernde Daten (Dateien und DVZ)

Schalter:

c : Überschreiben der Daten auf dem Datenträger

r : Anhängen an die Daten auf der Diskette

x : Einlesen der Daten vom Datenträger

t : Ausgeben eines Inhaltsverzeichnisses des Datenträgers

v : erweiterte Protokollierung der bearbeiteten Daten

f Gerätedatei : Spezifikation des Datenträgers;
für "Gerätedatei":

/dev/fl2 = Diskettenlaufwerk ("floppy 2")

/dev/rts0 = Magnetbandkassettenlaufwerk (V5.2x)

/dev/tape = Magnetbandkassettenlaufwerk (V5.40)

Beachten Sie wiederum, daß der Schalter "-r" bei Magnetbandkassetten nicht anwendbar ist.

Beispiele und Erläuterungen

Sicherung der Pfade "/usr1/texte" und "/usr2/prog" auf Magnetbandkassette am Rechner "MX303" unter der Kennung "sicher", wobei die auf der Kassette vorhandenen Daten überschrieben werden sollen und die gesicherten Daten zu protokollieren sind:

```
rtar MX303 -l sicher -cvf /dev/rts0 /usr1/texte /usr2/prog
```

Sicherung der Dateien "klausur" und "vertrag" auf Diskette (ferner Rechner: "vertrieb", ferne Kennung: "argo"; die Daten sollen an den Disketteninhalt angehängt werden):

```
rtar vertrieb -l argo -rf /dev/fl2 klausur vertrag
```

Einlesen aller Daten von der Magnetbandkassette am Rechner "Mainz" in das aktuelle Dateiverzeichnis (mit erweiterter Protokollierung; ferne Kennung: "gast"):

```
rtar Mainz -l gast -xvf /dev/rts0
```

ruptime — Anzeigen der Zustände der Rechner im Netz

ruptime

Der Befehl zeigt an, ob die Rechner im Netz eingeschaltet sind, seit wann sie aktiv/inaktiv sind, wieviele Benutzer daran arbeiten und inwieweit die Rechner ausgelastet sind.

Erläuterung anhand einer Beispiel-Ausgabe:

```
mx3003   up 3+04:01,  1 user,  load 0.06, 0.01, 0.00
mx3004 down    3:57
mx3005   up 1+03:35, 10 users, load 0.01, 0.00, 0.00
mx3006   up 1+03:19,  1 user,  load 0.29, 0.13, 0.03
```

Im Beispiel wird deutlich, daß der Rechner "mx3003" seit 3 Tagen ("3+) und etwa 4 Stunden ("04:01") aktiv ist. Gegenwärtig arbeitet ein Benutzer daran. Der Rechner "mx3004" ist seit etwa 4 Stunden inaktiv.
In den letzten Spalten ist die durchschnittliche Rechnerauslastung angezeigt. Die drei Zahlen gelten für die letzten 5, 10 und 15 Minuten und zeigen die durchschnittliche Anzahl der Prozesse, die ausführbereit sind und um den Prozessor konkurrieren.

rwho

rwho (-a) **Aktive Sitzungen im Netz anzeigen**

Es wird angezeigt, welche Kennungen im Netz seit wann an welchem Bildschirm arbeiten. Zusätzlich wird angezeigt, seit wann keine Eingabe mehr getätigt wurde.

Schalter:
-a: Auch Kennungen anzeigen, die seit mehr als einer Stunde keine Eingaben mehr am Bildschirm getätigt haben.

Erläuterung anhand einer Beispiel-Ausgabe:

```
admin   mx3006:tty01 Aug  2 12:25
brand   mx3003:tty03 Aug  2 09:03
gast    mx3006:tty03 Aug  2 12:24 :03
gerd    mx3006:tty00 Aug  2 12:00
```

In diesem Beispiel wird deutlich, daß insgesamt vier Kennungen im Netz aktiv sind. Der Benutzer "gast" arbeitet am Rechner "mx3006" (Terminal 03) und hat sich am 2. August um 12:24 Uhr angemeldet. Er hat seit 3 Minuten keine Eingabe mehr getätigt.

talk Kennung(@Host)	Direkte Unterhaltung am Bildschirm

talk

"talk" ermöglicht einen direkten schriftlichen Informationsaustausch am Bildschirm zwischen zwei Benutzern. Der Befehl ist auch innerhalb eines LAN einsetzbar.

Wenn "Kennung" nicht angemeldet ist, gibt der Befehl eine Fehlermeldung aus. Nach dem Aufruf bekommt der angesprochene Kommunikationspartner eine Nachricht auf seinem Bildschirm angezeigt, die ihm mitteilt, daß ein bestimmter Benutzer (die Kennung, die den Befehl aufgerufen hat) mit ihm kommunizieren will. Die Meldung wird in etwa zweiminütigem Abstand nochmals abgesetzt, bis die angerufene Kennung ebenfalls mit einem "talk"-Befehl antwortet. Dabei muß die Kennung und ggf. der Rechnername des Anrufers angegeben werden.

Anruf

Antwort

Der Bildschirm wird bei beiden Kommunikationspartnern in zwei Teile getrennt, wobei im oberen Teil die eigenen Eingaben und im unteren Teil die Eingaben des Partners zu sehen sind. Alles, was man eingibt, wenn die Verbindung steht, ist auf beiden Bildschirmen zu sehen. Es ist sogar möglich, daß beide Partner gleichzeitig Eingaben tätigen.

Sie können nur mit der <BACKSPACE>-Taste korrigieren, nicht aber z. B. mit den Pfeiltasten. Man beendet die Unterhaltung, indem man <DEL> drückt.

talk beenden

Beispiele und Erläuterungen

Benutzerin "marion" ruft Benutzer "baer" auf dem lokalen Rechner:

```
talk baer
```

Benutzer "baer" antwortet mit "talk marion".

Benutzerin "marion" am Rechner MM1 ruft Benutzerin "maria" auf dem fernen Rechner MM2:

talk maria@MM2

Benutzerin "maria" antwortet mit "talk marion@MM1".

---

write

write Kennung < Datei **Bildschirmnachricht an eine Kennung senden**

Der Inhalt von "Datei" wird auf den Bildschirm der spezifizierten Kennung gesendet, soweit der Adressat nicht mit dem Befehl "mesg n" die Ausgabe von Meldungen auf seinem Terminal verboten hat. "write" kann nicht netzweit benutzt werden.

Beispiele und Erläuterungen

Den Inhalt der Datei "frohe.ostern" auf den Bildschirm der Kennung "karin" schicken:

write karin < frohe.ostern

3.6 Kommandos für die Systemverwaltung

In diesem Kapitel werden einige Kommandos beschrieben, die **nur für die Systemverwaltung** (Kennungen "root" und "admin") relevant sind und elementare Aufgaben wie z. B. Datensicherung, Speicherplatzüberwachung und Datensicherheit betreffen.

Systemverwaltung

chown Kennung Pfad (...)	Eigentümer ändern

chown

Dieses Kommando ändert den Eigentümer von Verzeichniseinträgen (change owner).

Beispiele und Erläuterungen

Den Benutzer "gast" zum Eigentümer der Datei "Z3" machen:

```
chown gast Z3
```

Der Kennung "lucifer" Eigentümerrechte für alle Dateien und Dateiverzeichnisse der aktuellen und der darunterliegenden Verzeichnisebene vergeben (Vorsicht: Bei sehr vielen Einträgen könnte die Kommandozeile zu lang werden):

```
chown lucifer * */*
```

/etc/dump (Stufe)Schalter Partition	Gesamte Partition sichern

/etc/dump

Die gesamte angegebene Partition wird auf Magnetbandkassette (MBK) gesichert. Je nach angegebener Stufe kann man das gesamte oder nur die seit der letzten Sicherung neuen bzw. veränderten Teile des Dateisystems sichern. Die Datensicherung sollte im Ein-Benutzer-Betrieb (vgl. Kommando "/etc/shutdown") erfolgen. Die Sicherung kann mit dem Befehl "/etc/restore" wieder zurückgeschrieben werden.

Stufe: Die Sicherungsstufe gibt an, welche Daten zu sichern sind. Sie variiert von 0 bis 9. "0" bedeutet immer, daß das gesamte Dateisystem gesichert wird, und zwar mit sämtlichen Einträgen. Wenn man z. B. die Stufe 5 wählt, prüft das System, welche Daten seit der letzten Sicherung, die kleiner als die Stufe 5 war (z. B. 3), neu entstanden sind oder verändert wurden; nur diese werden auf dem Datenträger gesichert. Man spart auf diese Weise wertvollen Speicherplatz und Zeit.
Es empfiehlt sich also, auf Stufe 0 anzufangen und dann "hochzuzählen", so daß das System im Fehlerfall schrittweise restauriert werden kann (mit dem "restore"-Befehl).

Schalter: fu[c(s 3600)] /dev/rts0

f /dev/rts0	Hier wird die Gerätedatei für den Datenträger angegeben (in diesem Fall: MBK-Laufwerk).
u	Die Datei "/etc/dumpdates" wird aktualisiert, in der die Sicherungszeitpunkte und -stufen gemerkt werden (Standardangabe).
c	Gibt an, daß eine 45-MB-Magnetbandkassette als Datenträger verwendet wird.
s 3600	Wenn eine 60-MB-Kassette zur Sicherung benutzt werden soll, muß mit diesem Schalter die Länge des Sicherungsbandes angegeben werden (3600 Fuß).

Partition: Für das Dateisystem muß der Name der Gerätedatei angegeben werden, also z. B. "/dev/is0h" für "/usr1". Die Zuordnung der logischen und physikalischen Namen können Sie mit Hilfe des "df"-Kommandos herausfinden.

Daten-reorganisation mit dump und restore

Eine wichtige Anwendung ist neben der Datensicherung die Datenreorganisation mit "dump" und "restore":
Die Daten auf der Festplatte sind nach einer längeren Betriebszeit "ungeordnet", d. h. die physikalische Organisation weicht von der logischen Adressierung immer stärker

ab. Die dadurch entstehende Verlängerung der Zugriffszeiten beeinträchtigt die Performance des Systems. Deshalb ist es empfehlenswert, in regelmäßigen Abständen eine Datenreorganisation durchzuführen, die mit Hilfe der Kommandos "dump" und "restore" realisiert werden kann:
Man erstellt mit "dump" eine Sicherung der Stufe 0 (alles) und spielt sie mit "restore" wieder ein. Sicherheitshalber sollte man vor dem Wiedereinspielen die Daten der Partition mit dem Befehl "rm -r" löschen.

Beispiele und Erläuterungen

Vorgehen beim "dump"-Befehl:

Beispiel: Datensicherung mit dump

1. Login unter "root" an der Konsole.
2. Geben Sie nacheinander folgende Kommandos ein:

```
/etc/shutdown '+3' 'Datensicherung'
(Nach 3 Minuten:)
/etc/umount -a
/etc/fsck -y
```

Das System geht nun in 3 Minuten in den Ein-Benutzer-Betrieb, und alle Partitionen werden logisch vom Sinix-Dateisystem "abgehängt", damit sie gesichert werden können. Sicherheitshalber wird ein Konsistenztest durchgeführt, um die logische Richtigkeit aller Dateisysteme zu gewährleisten. Es kann sonst zu Problemen bei der Rückspeicherung kommen.

Legen Sie nun die 60-MB-Magnetband-Kassette ein.

Jetzt wird das "dump"-Kommando eingegeben. Mit dem folgenden Befehl wird eine vollständige Sicherung des Dateisystems "/dev/is0h" durchgeführt:

```
/etc/dump 0fus 3600 /dev/rts0 /dev/is0h
```

Erklärung der Komponenten:
0: alles sichern;
f: Gerätedatei des Datenträgers, hier: Magnetbandkassette;
u: Eintrag der Sicherung in die Datei /etc/dumpdates;
s 3600: 60-MB-Magnetband-Kassette als Sicherungsmedium;
/dev/rts0: Gerätedatei des Magnetbandkassettenlaufwerks;
/dev/is0h: Gerätedatei-Angabe der zu sichernden Partition, z. B.: "/dev/is0h" für "/usr1"

Während der Sicherung gibt "dump" Meldungen aus, die auf die Anzahl der benötigten Datenträger und die Dauer der Sicherung hinweisen. Gegebenenfalls wird ein neuer Datenträger angefordert.

Wenn Sie nun den folgenden Befehl eingeben und danach auf <ENDE> drücken, lädt das System wieder den Mehrbenutzerbetrieb:

/etc/mount -a

Wie Sie die Sicherung wieder einspielen, wird in den Beispielen zum "restore"-Kommando geschildert.

Erstellung einer Sicherung der Partition "/usr1" (Stufe 4) mit einer 60-MB-Kassette (die "Vorarbeiten" sind die gleichen wie im ersten Beispiel):

/etc/dump 4fus 3600 /dev/rts0

/etc/fsck

/etc/fsck (-[yn]) (Dateisystem) **Konsistenztest von Dateisystemen**

Das Kommando findet und behebt Inkonsistenzen (Widersprüche) in Dateisystemen, z. B. falsche Blockungen, fehlende Datei-Indizes (file system check). Für "Dateisystem" muß der Name der Gerätedatei der jeweiligen Plattenpartition angegeben werden (mit dem "df"-Kommando herauszufinden), ansonsten wird das gesamte Dateisystem bearbeitet.

Schalter:

-y : Alle Fragen, die während des Ablaufs gestellt werden, sollen mit "ja" ("yes") beantwortet werden. (Bei jeder einzelnen Inkonstistenz müßte sonst bestätigt werden.)

-n : Alle Fragen sind mit "nein" ("no") zu beantworten.

Beispiele und Erläuterungen

Es ist zu empfehlen, das Kommando im Ein-Benutzer-Betrieb an der Konsole abzusetzen. Man geht dann wie folgt vor:

/etc/shutdown '+5' 'System-Operation'
(Nachdem die 5 Minuten zu Ende sind:)
/etc/umount -a
/etc/fsck -y

Wenn Sie nun den folgenden Befehl eingeben und danach auf <ENDE> drücken, lädt das System wieder den Mehrbenutzerbetrieb:

/etc/mount -a

---

quot -f Dateisystem **Plattenbelegung anzeigen** quot -f

Anzeigen der Belegung der Plattenbereiche pro Benutzer, um so Benutzer herauszufinden, die zuviel Platz in Anspruch nehmen. Als "Dateisystem" ist die Gerätedatei der jeweiligen Plattenpartition einzutragen. Wenn das "df"-Kommando (siehe dort) z. B. für das Dateisystem "/usr1" die Gerätedatei "/dev/is0h" anzeigt, müssen Sie diesen Pfadnamen im Kommando angeben.
Es ist zu empfehlen, dieses Kommando in Verbindung mit "pg" bzw. "more" aufzurufen, z. B.: quot -f /dev/is0h ¦ pg

quot ¦ sie more
quot ¦ att pg

Beispiel für die Ausgabe des quot-Kommandos:

Beispiel-Ausgabe

```
/dev/is0h:
15970   1344   gast2
 5846    969   lorenz
 4848    185   lis
  826    189   gast
  662    167   lucifer
  304     28   root
  161     80   inform
    1      1   kolibri
```

1. Spalte: Anzahl der belegten Blöcke (à 512 Bytes)
2. Spalte: Anzahl der Dateien
3. Spalte: Eigentümer dieser Dateien

Die Anzahl der von einer Kennung belegten Blöcke ist die entscheidende Kennzahl für die Speicherplatzbelegung. Hier ist festzustellen, daß z. B. der Benutzer "gast2" zuviel Speicherplatz auf der Festplatte einnimmt (soweit er keine Sonderaufgaben hat wie z. B. ein Software-Entwickler).

/etc/restore

/etc/restore Schalter (Pfade) | **Wiedereinspielen einer "dump"-Sicherung**

Als Gegenstück zum "dump"-Kommando restauriert "restore" mit "dump" erstellte Sicherungen von Plattenpartitionen. Das Wiedereinspielen sollte im Ein-Benutzer-Betrieb erfolgen.

Schalter: [rRxtv]f /dev/rts0

r : Einlesen des gesamten Bandinhalts in das aktuelle DVZ (vollständige Restauration)

R : Wie "r", Folgebänder werden hier aber automatisch angefordert.

x ... Pfade : Nur die spezifizierten Pfade werden vom Band in das aktuelle DVZ eingelesen. Bei mehreren Bändern wird empfohlen, mit der höchsten (!) Bandnummer zu beginnen, damit nicht sämtliche (umfangreicheren) Sicherungen der untersten Stufe abgesucht werden müssen.

t : Ausgabe eines Inhaltsverzeichnisses der Daten auf dem Band (es wird also nichts zurückgeschrieben). Ohne Pfadspezifikation wird der gesamte Inhalt ausgegeben.

v : Die Namen aller bearbeiteten Verzeichniseinträge werden protokolliert.

f /dev/rts0 : Angabe der Gerätedatei des Datenträgers (hier: Magnetbandkassettenlaufwerk). Ohne den Schalter wird auf das ½-Zoll-MB-Laufwerk zugegriffen.

Beispiele und Erläuterungen

Beispiel:
Daten wieder einlesen mit restore

Vorgehen beim "restore"-Befehl:

Das folgende Beispiel liest die "dump"-Sicherung der Stufe 0 des Beispiels beim Befehl "/etc/dump" ein (siehe dort). Wenn Sie sich nicht im Ein-Benutzer-Betrieb befinden, müssen Sie noch die folgenden Kommandos eingeben:

```
/etc/shutdown '+5' 'System-Operation'
(Nach 5 Minuten:)
/etc/umount -a
/etc/fsck -y
```

Legen Sie den ersten "dump"-Datenträger ein, und geben Sie die folgenden Kommandos (an der Konsole!) ein:

rm -r /usr1/* /usr1/.[!.]* /usr1/..?*

(Löschen der Plattendaten, die zurückzuspeichern sind.)

/etc/restore rf /dev/rts0 /usr1

Erklärung der Komponenten:
r : Vollständiges Wiedereinlesen der Daten;
f /dev/rts0 : Angabe der Gerätedatei (/dev/rts0 für MB-Kassette);
/usr1 : Wiederherzustellende Partition;

Der Befehl läuft erheblich länger als das "dump"-Kommando. Wenn Sie nach der Beendigung des Kommandos den folgenden Befehl eingeben und danach auf <ENDE> drücken, wird vom System wieder der Mehr-Benutzer-Betrieb geladen:

/etc/mount -a

Weitere Beispiele:
Einlesen einer Sicherung (die Stufe spielt keine Rolle), die auf mehrere Bänder aufgeteilt ist, wobei die bearbeiteten Daten protokolliert werden sollen:

/etc/restore Rvf /dev/rts0 /usr1

Selektives Einlesen des DVZ "/usr1/golf/vertraege":

/etc/restore xvf /dev/rts0 /usr1/golf/vertraege

/usr/etc/secure (Schalter) — **Sicherheits-Check des Systems** — /usr/etc/secure

Das Programm untersucht das System auf folgende Informationen und gibt einen Bericht aus:

- Kennungen ohne Kennwort,
- Pseudo-Kennungen (die keinen Login erlauben),
- Kennungen mit doppelter Benutzer-Nummer,
- Kennungen ohne Login-Dateiverzeichnis,
- Benutzer, die seit mehr als 30 Tagen nicht mehr angemeldet waren (den Login nicht durchgeführt haben),
- Dateien mit gesetztem "s-Bit",
- für jeden beschreibbare Dateiverzeichnisse,
- normale Dateien im Dateiverzeichnis "/dev" (das alle Gerätedateien enthält),

- Geräte-Dateien außerhalb von "/dev",
- für jeden beschreibbare Gerätedateien in "/dev" (z. B. wäre ein genereller Schreibzugriff auf die Gerätedateien der Plattenpartitionen für die Datensicherheit fatal),
- Kopien von "/bin/sh" (z. B. könnten privilegierte Versionen versteckt existieren, die eine Systemverwalter-Shell erzeugen),
- Steuerzeichen-Folgen in Post-Dokumenten.

Schalter:

-h : (Kurze) Hilfe-Informationen ausgeben

/etc/shutdown

/etc/shutdown (-h) Zeitangabe

Rechner manuell ausschalten bzw. Ein-Benutzer-Betrieb aufrufen

Systemstopp

Ein-Benutzer-Betrieb

Das Kommando ist relevant, wenn der Systemverwalter den Rechner schnell ausschalten muß. Man kann dies durch einen Login unter der Kennung "shutdown" erreichen oder durch die Eingabe des Kommandos "/etc/shutdown -h Zeitangabe" unter der Kennung "admin" bzw. "root". Eine weitere Anwendung ist die Ausschaltung des Mehrbenutzerbetriebs: Wenn Sie unter "root" oder "admin" an der Konsole angemeldet sind, gelangen Sie nach Eingabe des Kommandos (ohne den Schalter "-h" in den **Ein-Benutzer-Betrieb**, d. h., das System ist nicht mehr im Mehrplatzbetrieb, und Sie arbeiten als einziger Benutzer. Mögliche Zeitangaben sind die folgenden:

+Zahl : In "Zahl" Minuten wird der Rechner softwaregesteuert ausgeschaltet ("heruntergefahren") bzw. der Ein-Benutzer-Betrieb aufgerufen. Die noch angemeldeten Benutzer bekommen jede Minute eine entsprechende Meldung.

now : Die gewünschte Aktion wird sofort durchgeführt.

att

wall Datei	**Absenden einer Bildschirmnachricht an alle am System angeschlossenen Benutzer**	wall

Alle Benutzer erhalten den Text, der in der Datei steht, sofort am Bildschirm angezeigt. (Für "private" Bildschirmmitteilungen an einzelne Benutzer verwendet man das "write"-Kommando.)

3.7 Sonderfunktionen der Shell (II)

3.7.1 Steuersequenzen für die Bildschirmdarstellung

Die Zeichen, mit denen Sie Kommandos eingeben, sind normale, druckbare Zeichen. Es gibt jedoch auch Zeichen(-folgen), die nicht auf diese Art und Weise sichtbar sind. Ein Beispiel ist die <RETURN>-Taste: Sie erzeugt eigentlich das unsichtbare Zeichen "Neue Zeile", das die Shell anweist, den Cursor an den Anfang der nächsten Bildschirmzeile zu versetzen und die eingegebene Zeile als Kommando

Steuersequenzen

zu interpretieren. Mit sogenannten Bildschirm-Steuersequenzen kann u. a. die Ausgabe der Daten am Bildschirm und die Bildschirmanzeige selbst beeinflußt werden. Im folgenden sollen nur einige kurz erläutert werden, die z. B. bei Fehlerfällen hilfreich sein können.

ESC C	Rücksetzen der Bildschirmausgabe auf die Ursprungswerte (<ESC>-Taste drücken, dann "C" und mehrmals <RETURN>)
CTRL-O	Ausschalten des Graphik-Zeichensatzes (<CTRL>-Taste drücken, festhalten und den Buchstaben "O", dann <RETURN>)
CTRL-S	Bildschirmausgabe anhalten
CTRL-Q	Bildschirmausgabe nach CTRL-S wieder fortsetzen

3.7.2 Apostrophierungsmechanismen

Apostrophierung

" ' `

Durch die Angabe von verschiedenen Apostrophierungszeichen werden Zeichenketten von der Shell unterschiedlich interpretiert.

"..."

"...": Nur die Zeichen $, ' und ` behalten ihre Sonderbedeutung. Variablen werden also durch ihre Werte ersetzt, da sie mit "$" beginnen.

'...'

'...': Der Inhalt (auch andere Apostrophierungen) wird als eine normale Zeichenkette aufgefaßt.

\`...\`: (Gravis) Kommandosubstitution, d. h., der Inhalt wird als Kommando ausgeführt, dessen Ergebnis den ursprünglichen Inhalt der Anführungszeichen ersetzt. \`...\`

Beispiele und Erläuterungen

Eingabe	**Ausgabe**
echo "Warten..."	Warten...
echo "Hallo, <$USER!>"	Hallo, <gast!> (wenn von "gast" aufgerufen)
echo 'Hallo, $USER!'	Hallo, $USER!
echo "Datum: \`date\`"	Datum: Di, 21.Mai 1991,15:00 MEZ
echo '\`date\`'	\`date\`

Eine nützliche Anwendung der Kommandosubstitution besteht im Merken des aktuellen Dateiverzeichnisses. Hier ein Beispiel in Form von Ein- und Ausgaben in der Shell:

```
$ pwd
/usr/lib/hit/original/printcap
         (das aktuelle Dateiverzeichnis wird ausgegeben)
$ dvz=`pwd`
         (die Variable "$dvz" wird mit dem Pfadnamen des
         aktuellen DVZ besetzt)
$ cd /usr
         (Wechsel zu einem anderen Dateiverzeichnis)
$ cd $dvz
         (Wechsel zurück zum gemerkten Dateiverzeichnis)
```

3.7.3 Ende-Status von Kommandos/Prozeduren

In Sinix bzw. Unix wird durch den sogenannten "Ende-Status" angezeigt, ob ein Kommando oder eine Prozedur erfolgreich abgelaufen ist. Der Ende-Status "0" bedeutet, daß bei der Ausführung des Befehls **kein** Fehler aufgetreten ist. Ein Ende-Status ungleich "0" (meistens "1") zeigt einen Fehler an. Er wird in der Systemvariablen "$?" nach Ablauf jedes Kommandos festgehalten und kann so z. B. in einer Prozedur ausgewertet werden. Ende-Status

&&, ¦¦

Mit Hilfe der Zeichen "&&" und "¦¦" kann man den Ende-Status automatisch auswerten. Syntax:

Kommando1/Prozedur1 [&& ¦¦] Kommando2/Prozedur2

Wenn "&&" angegeben ist, wird das zweite Kommando nur bei erfolgreicher Ausführung des ersten Kommandos ausgeführt (Ende-Status "0"). Dagegen wird bei einem Ende-Status ungleich "0" (nicht erfolgreich) das zweite Kommando nur ausgeführt, wenn "¦¦" zwischen den Befehlen steht.

Beispiele und Erläuterungen

Geben Sie nacheinander die folgenden Befehle ein:

```
ls
echo $?
lx
echo $?
```

Das erste Kommando ("ls") ist richtig und müßte somit den Ende-Status "0" aufweisen. Durch die ersten beiden Befehle wird also zuerst der Inhalt des aktuellen DVZ gelistet, dann eine Null ausgegeben. Das folgende Kommando ist falsch (soweit keine Prozedur namens "lx" existiert). Die Meldungen der letzten beiden Eingaben lauten also ("1" ist der Ende-Status des fehlerhaften Kommandos):

```
lx: nicht gefunden          (bzw. "lx: not found")
1
```

Wenn die Datenbankabfrage mit dem "isql"-Kommando erfolgreich abläuft, soll eine Erfolgsmeldung ausgegeben werden:

```
isql DB pers.sql > outdat && \
echo "Die Abfrage der Personaldatenbank war erfolgreich!"
```

Für das gleiche Kommando soll im Fehlerfall der Ende-Status ausgegeben werden:

```
isql DB pers.sql > outdat ¦¦ \
echo "Fehler bei der Datenbankabfrage; Ende-Status: $?"
```

3.7.4 Die Datei "$HOME/.profile"

Wenn man im eigenen Stamm-Dateiverzeichnis eine Datei namens ".profile" anlegt, wird diese Datei bei jedem Login automatisch als Shell-Prozedur ausgeführt. Da die Systemvariable "$HOME" immer den absoluten Pfadnamen Ihrer Home-Directory enthält, kann man die Datei allgemein mit "$HOME/.profile" bezeichnen.

Der Nutzen dieser Login-Start-Prozedur zeigt sich z. B. bei INFORMIX: Hier können Sie bequem (mit der Variable "$DBPATH") den Datenbankpfad festlegen oder (mit "$DBPRINT") ein automatisches Druckprogramm für Formatprogramme. Auch andere Software-Produkte (z. B. SIPLAN, OCIS, HIT) lassen sich über Variablen konfigurieren, die empfehlenswerterweise in der Datei ".profile" definiert werden.

Beispiele und Erläuterungen

Sie können in diese Datei auch Befehle schreiben, die Ihnen beispielsweise wichtige Informationen vermitteln (z. B. "df", "who", "du -s"). Auch der Ausbau zu einer komplexen Shell-Prozedur ist denkbar.

Die Definition von Variablen könnte wie folgt aussehen:

```
DBBPATH=/usr1/dbsys/DB
# Datenbank-Pfad-Variable
DBPRINT='pr -172 -h "INFORMIX-OUTPUT" ¦ lpr -dt -pb2'
# Druck-Variable
DBTMP=$HOME/tmp
# Pfad zur Ablage temporärer INFORMIX-Daten
export DBPATH DBPRINT DBTMP
# Bekanntmachung der Variablen für alle Folgeprozesse
```

3.8 Rückblick auf die Vorgänge bei der Interpretation

Nachdem Sie in den vorigen Kapiteln sowohl die wichtigsten "theoretischen" Grundlagen als auch einen repräsentativen Querschnitt von Kommandos und Sonderfunktionen kennengelernt haben, scheint es angebracht, nochmals eine kurze zusammenfassende Erklärung der Aktionen des Kommando-Interpreters zu geben: Was macht die Shell, sobald Sie ein Kommando eingeben und "abschicken"?

Gehen wir von einer umfangreichen Kommandozeile aus, die vom Benutzer "gast" im Dateiverzeichnis "/usr/gast/dvz" eingeben wird. Es sollen alle Einträge des aktuellen DVZ, die nicht mit einem Punkt beginnen, mit allen Unterdateiverzeichnissen gesichert werden. ("cpio -o" erhält die absoluten Pfadnamen als Eingabe über die Pipeline.) Das Ergebnis ist eine Sicherungsdatei, die im Stamm-Dateiverzeichnis des aufrufenden Benutzers abgelegt werden soll (Name: "SICHER.kennung", hier also "SICHER.gast"):

```
find `pwd`/* -print ¦ cpio -o > $HOME/SICHER.$USER
```

Erläuterungen:

Kommando-substitution

1. Die Shell führt zunächst das Kommando "pwd" aus (Kommandosubstitution mit "`"). Die Zeile sieht dann wie folgt aus:

   ```
   find /usr/gast/dvz/* -print ...
   ```

Dateinamen-generierung

2. Nun wird der Stern durch alle Eintragsnamen in diesem DVZ ersetzt (Dateinamengenerierung). Ergebnis:

   ```
   find /usr/gast/dvz/DB /usr/gast/dvz/dat ... -print
   ```

Verarbeitung

3. Das Kommando "find" hat nun die endgültige Form und wird von der Shell an den Systemkern zur Ausführung übergeben. Die vom Kern zurückgegebene Ausgabe des Befehls wird über die Pipeline direkt an den Befehl "cpio" weitergegeben und könnte ausschnittsweise z. B. wie folgt aussehen:

```
/usr/gast/dvz/DB
/usr/gast/dvz/dat
/usr/gast/dvz/DB/maske.per
/usr/gast/dvz/DB/liste.ace
```

4. Durch das Zeichen ">" weiß die Shell, daß das Verarbeitungsergebnis des "cpio"-Befehls in eine Datei umgeleitet werden soll. Hierzu werden die beiden Variablen, die im Dateinamen enthalten sind, zunächst ersetzt, dann wird die entsprechende Datei angelegt: — *Umleitung* — *Variablenersetzung*

```
... ¦ cpio -o > /usr/gast/SICHER.gast
```

5. Im letzten Schritt wird die Datei mit der Ausgabe des "cpio"-Kommandos gefüllt. Sämtliche internen Verarbeitungsvorgänge (z. B. Reservierung von Rechenzeit, Prozeßerzeugung) werden automatisch durchgeführt. Den Abschluß bildet die Besetzung der Systemvariable "$?", die den Ende-Status der Kommandozeile enthält (sie hat den Wert "0", wenn kein Fehler aufgetreten ist). — *Ausgabe* — *Ende-Status*

4. Die Shell als Programmiersprache

4.1 Funktion und Bedeutung von Prozeduren

Prozeduren

Eine Shell-Prozedur (kurz: Prozedur) ist eine Kommandodatei, d. h. eine Datei, die eine Ansammlung von Shell-Kommandos enthält und wie ein Programm ausgeführt werden kann. Der Interpreter liest seine Eingaben dann aus dieser Datei.

Unix bzw. Sinix bietet hierbei die Vorteile, daß diese Kommandos für einen Kommando-Interpreter **sehr schnell** ausgeführt werden. Außerdem ist keine Kompilierung wie bei "richtigen" Programmiersprachen notwendig. Auf der anderen Seite stehen durch **Ablaufstrukturen** (z. B. zur Bildung von Schleifen) praktisch die Grundfunktionen einer normalen Programmiersprache zur Verfügung.

Zweck von Prozeduren

Shell-Prozeduren haben vor allem folgende **Funktionen**:

- Die Automatisierung von komplexen Aktionen,
- die Einbindung von programmierten Anwendungen,
- die Vereinfachung und benutzerfreundlichere Gestaltung von Anwendungen und häufig benutzten Aktionen sowie
- die Verhinderung von Fehlern, indem Kommandos bzw. Aktionen sicherer gestaltet werden.

4.2 Eine Prozedur schreiben und aufrufen

Man erstellt **Prozeduren** mit Hilfe von ASCII-Editoren (z. B. mit dem CED), also Editoren, die ohne besondere Steuerzeichen arbeiten. (Mit dem Textsystem HIT erstellte Texte werden dagegen von der Shell nicht korrekt interpretiert.) Gehen Sie wie folgt vor:

Prozedur erstellen

1. **Schreiben einer Prozedur:**
   Wechseln Sie in die Shell und laden Sie eine Datei in den CED. Schreiben Sie dann die gewünschten Zeilen. Schreiben Sie z. B. in die Datei "proz1":

   ```
   echo "Hallo, $USER!"
   ```

2. **Aufruf der Prozedur**

   Erste Möglichkeit:

   Aufruf mit sh bzw. "."

   Es ist grundsätzlich immer möglich, eine eigene Prozedur mit dem Befehl "**sh**" aufzurufen. Sie erzeugen damit eine neue Subshell. Geben Sie also ein:

   ```
   sh proz1
   ```

   Wenn Sie die Prozedur innerhalb der aktuellen Shell ablaufen lassen wollen, müssen Sie anstelle von "sh" einen Punkt (".") eingeben:

   ```
   . proz1
   ```

   Dann sind Variablen, die in der Prozedur definiert werden, auch der akutellen (aufrufenden) Shell bekannt, was bei "sh" nicht der Fall ist.

   Zweite Möglichkeit:

   Aufruf mit Ausführberechtigung

   Wenn man nicht immer den "sh"-Befehl verwenden möchte, so macht man die Prozedur mit dem Befehl "**chmod**" ausführbar. Geben Sie ein ("u+x" vergibt Ausführerlaubnis für den Eigentümer der Datei):

   ```
   chmod u+x proz1
   ```

   Die Prozedur wird nun aufgerufen, indem in der Shell einfach ihr **Name** und **<RETURN>** eingegeben werden:

   ```
   proz1
   ```

4.3 Variablen in der Sinix-Shell (II)

Variablen
$

Sie wissen schon aus Abschnitt 2.4.5, daß man in Variablen veränderliche Daten speichert. Variablennamen beginnen mit dem Sonderzeichen "$". Von den drei grundsätzlichen Arten von Variablen sind die ersten beiden dort bereits erläutert worden:

Arten von Variablen

1. **(Vordefinierte) Standardvariablen (Systemvariablen)**
2. **Selbstdefinierte Variablen**
3. **Unbenannte Variablen (Stellungsparameter)**

export

In diesem Zusammenhang ist noch zu bemerken, daß durch den Aufruf einer Shell-Prozedur mit "sh" oder mit ihrem Dateinamen (soweit Ausführberechtigung gegeben ist) eine neue Subshell erzeugt wird. Die Kind-Shells Ihres aktuellen Prozesses kennen jedoch bis auf einige Systemvariablen nicht die Variablen der aktuellen Shell. Hierzu müssen Variablen "exportiert", d. h. den Unterprogrammen bekannt gemacht werden (vgl. Kommando "export" auf S. 107 f.).

Stellungs-variablen

Unbenannte Variablen (Stellungsparameter)
Diese Variablen werden statt durch eine Zeichenkette (z. B. "NAME" in "$NAME") durch eine einstellige Zahl gekennzeichnet: Bis zu 9 dieser Variablen können so direkt angesprochen werden: "$1" bis "$9". "$1" bedeutet also nicht Variable "1", sondern einfach nur "das erste Argument". Wertzuweisungen sind nicht wie bei selbstdefinierten Variablen möglich; dies geschieht sowohl bei der Anwendung des "set"-Kommandos als auch bei Prozeduraufrufen durch die Stellung (Reihenfolge) der Werte. Deswegen werden diese Variablen auch **Stellungsparameter** genannt. Hat eine Prozedur mehr als 9 Parameter, wird zwar für jeden Parameter eine Stellungsvariable besetzt - aber nur die ersten 9 sind direkt ansprechbar.

Die allgemeine Syntax (für eine Prozedur mit N Parametern, N > 9) lautet also folgendermaßen:

```
(sh) Prozedurname Par1 Par2 ... Par9 ... ParN          Prozeduraufruf
                  $1   $2   ... $9
```

Beim "set"-Kommando setzt man die Variablen wie folgt:

```
set Wert1 Wert2 ... Wert9 ... WertN                    set
    $1    $2    ... $9
```

$9 ist der letzte Stellungsparameter, der direkt angesprochen werden kann. Man kann jedoch mit dem Befehl "shift" Stellungsparameter "verschieben": Die Variable $1 wird mit dem Wert von $2 besetzt - ihr ursprünglicher Wert geht verloren -, $2 erhält den Wert von $3 usw., $9 erhält den Wert des ursprünglich 10. Arguments (falls vorhanden), usf. Mit $* werden alle Stellungsparameter angesprochen.

Beispiele und Erläuterungen

Geben Sie als Test die folgenden Kommandos ein:

```
cd /usr/admin/.benutzer
# Das DVZ .benutzer enthält alle Kennungsnamen
# in Form leerer Dateien
set `ls`
# Die unbenannten  Variablen werden mit den Namen der ein-
# zelnen Kennungen besetzt
echo 'Der Wert von $1 ist: ' $1
# Ausgabe des Wertes der ersten Variable usw.
echo 'Der Wert von $2 ist: ' $2
echo 'Der Wert von $3 ist: ' $3
echo 'Der Wert aller Variablen ($*) ist: ' $*
```

4.4 Kommandos zur Erstellung von Prozeduren

Unterstreichung in der Syntax

Die im folgenden kurz dargestellten Kommandos zur Erstellung von Prozeduren sind am besten mit Hilfe der Erläuterungen und Beispiele zu verstehen. Zusätzlich zur bisherigen Syntaxdarstellung sind die unveränderlichen Zeichen, die zu der jeweiligen Struktur gehören, unterstrichen. Die Beispiele zu den Kommandos werden durch Kommentarzeilen (eingeleitet durch "#") näher erläutert.

case

```
case Ausdruck in
  Muster1) Kommandoliste (;;
  Muster2) Kommandoliste )
  ...                    (;;
       *) Kommandoliste )
esac
```

Wertabhängige Verzweigung

Umgangssprachlich ausgedrückt, bedeutet die Syntax: Wenn "Ausdruck" (z. B. eine Variable oder die Ausgabe eines Kommandos) mit einem der angegebenen Muster übereinstimmt ("Muster1", z. B. ein Wert wie die Zeichenkette "mgast"), wird die dem Muster zugeordnete Kommandoliste ausgeführt. Danach fährt Sinix mit dem nächsten Befehl nach "esac" fort. Wird ein einzelner Stern ("*") als Muster angegeben, so gilt dieser Zweig für alle Möglichkeiten, die nicht durch die anderen Muster abgedeckt sind. Sterne wie auch andere Matchcodes können in den Mustern verwendet werden.

*: Sonst-Zweig

Die einzelnen musterspezifischen Kommandolisten werden durch zwei Semikola (";;") voneinander getrennt. Wenn diese Zeichen fehlen, erwartet der Kommando-Interpreter, daß das Ende der Verzweigung gekommen ist ("**esac**"). Die letzte Kommandoliste darf also nicht mit ";;" aufhören.

;;

esac

Man kann theoretisch beliebig viele Zeilen mit Mustern angegeben. Es können auch mehrere Muster für eine Aktion (Kommandoliste) angegeben werden, indem man die Werte mit dem Zeichen "¦" voneinander trennt (ohne Leerzeichen, und die Klammer am Ende gehört zur Syntax!):

Muster1¦Muster2¦Muster3) Muster1 ¦ Muster2

Die Verzweigung mit "case" wird angewendet, wenn sich eine Vielzahl verschiedener Alternativen bietet, die unterschiedliche Aktionen erfordern.

Beispiele und Erläuterungen

Je nachdem, wie die Variable "$USER" besetzt ist, werden der aufrufenden Kennung verschiedene Datenbankpfade (für INFORMIX) zugeordnet:

```
case $USER in
admin ¦ root)
  echo "Sie sind System- und Datenbank-Verwalter!"
  DBPATH=/usr1/DBverwalt ;;
leiter1 ¦ leiter2 ¦ leiter3)
  echo "Sie haben Zugriff auf die Abteilungsleiter-DB!"
  DBPATH=/usr1/Abt.Leiter ;;
vb* ¦ assi*)
# Passend wären z. B.: vb001, vb123, assi01, assist
  echo "Sie haben Zugriff auf die Vertriebsbeauftragten-
        und Assistenten-Datenbank!"
  DBPATH=/usr1/VB ;;
*) echo "Sie sind nicht befugt! Abbruch der Prozedur ..."
   exit 1
# "*" gilt für alle anderen Kennungen
esac

export DBPATH
# Die Variable wird auch in Subshells bekannt gemacht
```

Innerhalb eines Programmsystems werden den einzelnen Benutzern damit jeweils bestimmte Datenbankpfade zugeordnet.

---

echo Werte **Ausgabe von beliebigen Zeichen** echo

"echo" ist eigentlich für die Bildschirmausgabe von Zeichenketten gedacht, die "Werte" können aber auch in Dateien umgeleitet werden. Besonders wichtig ist der Befehl für

Prozeduren, weil damit Bildschirmmeldungen und Eingabeaufforderungen zu gestalten sind. Ohne "Werte" gibt "echo" eine leere Bildschirmzeile aus.
Der Befehl liegt für die Universen "sie" und "att" in zwei verschiedenen Versionen vor. Wenn man nicht will, daß der Cursor nach der Bildschirmausgabe in die nächste Zeile springt (das ist z. B. bei Eingabeaufforderungen interessant), so wendet man "echo" wie folgt an:

sie echo

sie echo -n Werte

oder

att echo

att echo "Werte \c"

(auch: att echo 'Werte \c')

Beispiele und Erläuterungen

Folgende Anwendungen des "echo"-Befehls sind besonders häufig:

- Eingabeaufforderungen: echo "Geben Sie etwas ein: \c"
- Bildschirmmeldungen: echo "Falsche Eingabe!"
- Ausgabe von Variablenwerten: echo $USER
- Ausgabe von Werten in eine Datei: echo `date` > Dat1

Im letzten Beispiel wird die Ausgabe des Kommandos "date" durch Kommandosubstitution in die Datei "Dat1" geschrieben.

Weitere Beispiele zu "echo" finden Sie in den Beispielen zu den Ablaufstrukturen. Zu bemerken ist noch, daß der "echo"-Befehl durch Apostrophierung über mehrere Zeilen gehen kann:

```
echo " ******* ACHTUNG *******
Wir weisen darauf hin, dass das System
in 5 Minuten heruntergefahren wird !!!"
```

exit (Ende-Status) Shell(-Prozedur) beenden exit

Der Befehl beendet Shell-Prozeduren augenblicklich. Den "Ende-Status" können Sie zwecks späterer Auswertung zusätzlich angeben ("0" heißt generell fehlerfreier Ablauf, andere Werte - z. B. "1" - Beendigung mit Fehler).

Beispiele und Erläuterungen

Wenn in den folgenden Beispielen die Konstruktion "[...]" verwendet wird, dann ist dies die "eingebaute" Version des "test"-Befehls (siehe dort).

Vorzeitiger Abbruch, wenn keine Eingabe gemacht wurde:

```
echo "Bitte geben Sie einen Wert ein: \c"
read antwort
# Hier ist die Variable "$antwort" vom Benutzer einzugeben

if [ x$antwort != x ]
then
  echo "Sie haben nichts eingegeben! Abbruch ..."
  exit 1
fi
```

Ausgabe eines INFORMIX-Ende-Status (die System-Variable "$?" enthält immer den Ende-Status):

```
echo "Bitte warten, der SQL-Befehl wird ausgeführt ..."
isql DB auswert.sql

if [ $? != 0 ]
then
  echo "Fehler-Status beim Ablauf der SQL-Datei: $?"
  exit 1
fi
```

export Variablenliste Variablen exportieren export

"Exportieren" bedeutet, daß die Variablen (die normalerweise nur für die aktuelle Shell bekannt sind) auch den Kind-Prozessen (Subshells) der aktuellen Shell bekannt gemacht werden. Die Variablennamen werden **ohne einleitendes "$"** angegeben.

Beispiele und Erläuterungen

Die Variablen "$DATEI" und "$DVZ" sollen nicht nur in der aktuellen Shell, sondern auch in von ihr aufgerufenen Subshells bzw. Prozeduren ansprechbar sein:

```
export DATEI DVZ
```

---

for

for Variablenname **in** Werteliste **do** Kommandoliste **done**	Schleifenbildung

Die Anzahl der durch Leerzeichen getrennten Werte in der "Werteliste" gibt an, wie oft die "Kommandoliste" ausgeführt wird (also z. B. 5 Durchläufe bei 5 Werten). Bei jedem Schleifendurchlauf wird "Variablenname" (ohne "$" anzugeben) mit dem nächsten Wert aus der Werteliste besetzt.

Beispiele und Erläuterungen

Die Dateien "brief", "skript1", "D1", "D2" und "D3-1" sollen jeweils angezeigt und dann interaktiv gelöscht werden:

```
for DATEI in brief skript1 D1 D2 D3-1
do
    pg $DATEI
    # Seitenweise Anzeige der Datei
    rm -i $DATEI
    # Löschen der Datei mit Abfrage
done
```

Allen Benutzern die Datei "nachricht" auf den Bildschirm senden, soweit die jeweilige Kennung angemeldet ist:

```
for KENNUNG in `(cd /usr/admin/.benutzer ; ls)`
# In dem DVZ ".benutzer" stehen die Namen aller Kennungen
# in Form von leeren Dateien. Wenn man nur `ls /usr/ad-
# min/.benutzer` angibt, erhält man statt der reinen Ken-
# nungsnamen (z. B. gast, zbv431) nur absolute Pfadnamen
# als Ausgabe (z. B. /usr/admin/.benutzer/gast). Durch die
# Apostrophierung mit "`" findet eine Kommando-Substitu-
# tion statt, d. h., in der Werteliste steht die Ausgabe
# des Kommandos.
do
    write $KENNUNG < nachricht
done
```

<u>if</u> Kommando <u>then</u> Kommandoliste (<u>else</u> Kommandoliste) <u>fi</u>	Bedingte Verzweigung	if

Wenn das "Kommando" korrekt - also mit Ende-Status "0" - ausgeführt wurde, wird die Kommandoliste ausgeführt. Andernfalls wird - soweit vorhanden - die Kommandoliste des "else"-Zweigs ausgeführt. Die "if"-Verzweigung wird meistens mit "[...]" ("test"-Kommando) kombiniert.

Beispiele und Erläuterungen

Wenn der Benutzer "ben01" existiert, bekommt er elektronische Post geschickt, andernfalls wird eine Fehlermeldung ausgegeben (unter "/usr/admin/.benutzer" sind alle Kennungsnamen in Form von leeren Dateien eingetragen):

```
echo "Bitte geben Sie den Namen einer Kennung ein: \c"
read KENNUNG

if [ -f /usr/admin/.benutzer/$KENNUNG ]
then
  mail $KENNUNG < nachricht
else
  echo "Fehler: Die Benutzerkennung existiert nicht!"
fi
```

read Variablenliste	Einlesen von Variablenwerten	read

Der Kommando-Interpreter hält an und wartet auf eine Benutzereingabe, mit der dann die angegebenen Variablen besetzt werden. Wenn mehrere Variablen eingelesen werden sollen, müssen die einzelnen Werte durch Leerzeichen getrennt werden. Die Namen der Variablen werden ohne "$" angegeben.
Im interaktiven Betrieb sollte der "read"-Befehl mit einer Eingabeaufforderung (mit "echo") kombiniert werden.

Beispiele und Erläuterungen

Einlesen der Variablen "$Vorname" und "$Name":

```
echo "Bitte geben Sie den Vornamen und den Namen ein: \c"
read Vorname Name
```

set

set Werteliste **Besetzung der Variablen $1 bis $9**

"set" versieht die unbenannten Variablen mit Werten der "Werteliste". Das können beliebig viele sein, jedoch sind nur die ersten neun mit "$1" bis "$9" direkt ansprechbar. Weil es "$10" nicht gibt, müssen Sie auf das Kommando "shift" (s. u.) zurückgreifen, um den Wert eines 10. Stellungsparameters abfragen zu können. - Die Gesamtheit der unbenannten Variablen wird mit der Systemvariable "$*" angesprochen.

$*

VORSICHT: Wenn keine Werte angegeben werden, gibt "set" alle aktuellen Umgebungsvariablen aus. Das kann bei Prozeduren unangenehm werden.

Der "set"-Befehl dient auch zum Setzen von Schaltern für die Shell (Ablaufsteuerung, z. B. Testläufe von Prozeduren). Nähere Informationen hierzu finden Sie im Handbuch "Betriebssystem Sinix V5.xx - Kommandos".

Beispiele und Erläuterungen

Die Auslastung der Plattenpartition "is0h" in Prozent ausgeben (Ausgabe z. B.: "9 %"):

```
set `df ¦ fgrep /dev/is0h`
echo $5
```

Hier wird die Ausgabe des Kommandos "df" nach dem Muster "/dev/is0h" durchsucht. Die durch Leerzeichen getrennten Komponenten der Ergebniszeile werden den Stellungsvariablen "$1" bis "$9" zugeordnet. An fünfter Stelle steht die gesuchte Prozentangabe.

Ein weiteres Beispiel zu "set" finden Sie bei den Ausführungen zum "while"-Kommando.

shift (Zahl) Stellungsparameter verschieben shift

Die unbenannten Variablen bzw. Stellungsparameter "$1" bis "$9" werden mit mit dem "set"-Kommando oder beim Aufruf von Prozeduren besetzt. Mit dem "shift"-Kommando werden Stellungsparameter um "Zahl" nach links verschoben.
Die Variable "$n" (n = 1, 2, ...) erhält also den Wert der Variablen "$n+Zahl". Wenn Sie für "Zahl" keinen Wert angeben, wird "1" angenommen.

Beispiele und Erläuterungen

Die unbenannten Variablen werden zunächst durch den "set"-Befehl mit Werten besetzt:

```
set a b c d e f g h i j k l m n
```

Die Stellungsparameter lauten damit wie folgt:

```
$1  $2  $3  $4  $5  $6  $7  $8  $9  Nicht ansprechbar
a   b   c   d   e   f   g   h   i   j   k   l   m   n
```

Die Eingabe des Kommandos "shift" hat nun die Konsequenz:

```
$1  $2  $3  $4  $5  $6  $7  $8  $9  Nicht ansprechbar
b   c   d   e   f   g   h   i   j   k   l   m   n
```

Hier hat sich die Zuordnung der Werte zu den einzelnen Parametern um eine Stelle verschoben. Die Eingabe von "shift 3" hat die Konsequenz:

```
$1  $2  $3  $4  $5  $6  $7  $8  $9  Nicht ansprechbar
e   f   g   h   i   j   k   l   m   n
```

Durch die Verschiebungen sind die Werte "a, b, c, d" unwiederbringlich verloren.

sleep

sleep Sekunden	Prozeß stillegen

Der aktuelle Prozeß wird für die angegebene Anzahl von Sekunden stillgelegt, d. h., in dieser Zeit werden keine Systemressourcen verbraucht. In Prozeduren bedeutet dies, daß erst nach Ablauf der spezifizierten Anzahl an Sekunden mit dem nächsten Befehl fortgefahren wird. Man kann das Kommando (und damit natürlich auch die Prozedur) mit der <DEL>-Taste abbrechen.

test

Der "test"-Befehl

Ende-Status

Der "test"-Befehl prüft Bedingungen. Ist eine Bedingung erfüllt ("wahr"), wird die Variable "$?", die den Ende-Status eines Kommandos enthält, auf "0" gesetzt; ist die Bedingung nicht erfüllt ("falsch"), erhält "$?" den Wert "1". Eine unmittelbar sichtbare Ausgabe erzeugt der "test"-Befehl nicht.

[...]

"test" wird hauptsächlich in Verbindung mit den Ablaufstrukturen "if" und "while" benutzt, die den Ende-Status direkt auswerten (vgl. Beispiele). Das Kommando kann auch in der Form eckiger Klammern ("[]") geschrieben werden ("eingebaute Schreibweise"), wobei einfach das Wort "test" weggelassen wird und die Argumente und Schalter durch Leerzeichen getrennt zwischen die Klammern geschrieben werden. Das Kommando läuft in dieser Form schneller ab. Es gibt grundsätzlich zwei Versionen des "test"-Befehls:

1. **test Schalter Argumente**
   bzw. [Schalter Argumente]

2. **test Ausdruck Operator Ausdruck**
   bzw. [Ausdruck Operator Ausdruck]

test Schalter Argument **Bedingungen prüfen** 1. test

Mit der Hilfe verschiedener Schalter können komfortabel bestimmte Bedingungen geprüft werden. Hier eine Auswahl der wichtigsten Schalter mit ihren Argumenten:

-d DVZ : Wahr, wenn "DVZ" als DVZ vorhanden ist
-f Datei : Wahr, wenn "Datei" als normale Datei existiert
-r Eintrag: Wahr, wenn die aufrufende Kennung Leseerlaubnis auf "Eintrag" (Datei oder DVZ) besitzt
-s Datei : Wahr, wenn "Datei" eine nicht leere Datei ist
-w Eintrag: Wahr, wenn die aufrufende Kennung Schreiberlaubnis auf "Eintrag" (Datei oder DVZ) besitzt

Zur Verknüpfung dieser Prüfungen siehe unten.

test Ausdruck1 Operator Ausdruck2 **Werte vergleichen** 2. test

Diese zweite Version des "test"-Befehls vergleicht zwei Ausdrücke miteinander. Ausdrücke sind Wertangaben, z. B. konstante Zeichenketten wie "gast", Variableninhalte wie "$USER" oder auch die Ausgabe einer Kommandosubstition wie bei "\`date\`". Wenn der Vergleich "stimmt", ist der Ende-Status "0", andernfalls "1".

Operatoren können sein:

Operatoren

= : Gleichheit
!= : Ungleichheit
-gt : größer als (greater than)
-ge : größer oder gleich (greater or equal)
-lt : kleiner als (less than)
-le : kleiner oder gleich (less or equal)

Verknüpfung

Die Prüfungen bzw. Bedingungen beider Versionen können negiert, verknüpft und zusammengefaßt werden:

! : Negierung (Umkehrung der Bedingung)

-a : Logisches UND (beide Bedingungen müssen erfüllt sein)

-o : Logisches ODER (nur eine Bedingung muß erfüllt sein)

() : Die Bedingungen in den Klammern sind zusammengefaßt

Beispiele und Erläuterungen

Beispiele für das "test"-Kommando finden sich in den Beispielen zu den Befehlen "exit", "if" und "while" . Dort wurde es immer in der eingebauten Schreibweise verwendet.

In der Shell können Sie das Kommando wie folgt testen. Prüfen, ob die Datei "/usr/gast/.profile" existiert:

```
test -f /usr/gast/.profile
```

Wenn Sie nun "echo $?" eingeben, wird "0" ausgegeben, wenn die Datei existiert, sonst "1". Das folgende Beispiel prüft, ob die Variable "$USER" mit "admin" besetzt ist ("Ist der aufrufende Benutzer Systemverwalter?"):

```
test $USER = admin
```

Weitere Beispiele:

[! -f /usr/gast/geheim] ist wahr, wenn die Datei "/usr/gast/geheim" nicht existiert.

[-w /etc/passwd -o ($USER = admin -o $USER = root)]
ist wahr, wenn der Aufrufende entweder Schreibrecht auf die Datei "/etc/passwd" besitzt oder wenn er unter einer der Kennungen "root" und "admin" arbeitet.

[x$1 != x] ist wahr, wenn die Variable "$1" nicht leer ist (wenn sie leer ist, ist "x = x"). Das "x" wurde hier als "Trick" zu Hilfe genommen, um Syntaxfehler zu vermeiden: Wenn die Variable leer ist, muß trotzdem auf beiden Seiten ein Wert stehen, damit eine Prüfung durchgeführt werden kann. Ohne die beiden "x" würde lediglich "!= " in den Klammern stehen, falls "$1" leer ist. Der "test"-Befehl hätte dann keine Werte zum Vergleichen, würde eine Fehlermeldung ausgeben und die Prozedur abbrechen.

while Kommando **do** Kommandoliste **done**	**Schleife ausführen**	while

Die Schleife wird ausgeführt, solange das Kommando fehlerfrei (mit Ende-Status "0") ausgeführt wird. Oft wird hier das "test"-Kommando verwendet, um Bedingungen zu prüfen.

Beispiele und Erläuterungen

Zum besseren Verständnis eine Beispielprozedur:

Postkörbe löschen

```
# mailclear.scr
#
# Löschen des Inhalts der Postkörbe der Benutzer (von
# admin aufzurufen); sinnvoll, weil die Postkörbe im Lauf
# der Zeit immer voller werden, wenn die Benutzer ihre
# Post nicht löschen.

set `ls /usr/spool/mail`
# Das Ergebnis der Auflistung des angegebenen Katalogs
# (absolute Pfadnamen der Postkörbe aller Benutzer) wird
# der Stellungsparameterliste zugeordnet.

while [ x$1 != x ]
do
# Schleife ausführen, solange die Variable $1 nicht leer
# ist.
# (Zur Konstruktion mit "x" vgl. das letzte Beispiel zum
# "test"-Befehl.)

> /usr/spool/mail/$1
# ">" als Befehl legt die Datei ohne Inhalt neu an, d. h.,
# sie überschreibt den alten Inhalt mit einer leeren
# Datei.

shift
# Parameter verschieben: $1 hat jetzt den Wert von $2.

done
# Schleifenende
echo "Die Prozedur wurde beendet."
```

4.5 Entwicklung einer Beispielprozedur

Das Schreiben von Shell-Prozeduren ist am besten zu lernen, wenn man ein konkretes praktisches Problem zu lösen hat und andere Problemlösungen kennt. Hier kann dieser weitläufige Themenkreis nur angedeutet werden. Fortgeschrittene und kommentierte Beispiel-Prozeduren finden Sie auf der beiliegenden Beispieldiskette.

Problemstellung

Formulierung der Problemstellung:
Um Benutzerfehler beim Löschen von Einträgen (Dateien und Dateiverzeichnisse) zu verhindern, soll diese Aktion benutzerfreundlich gestaltet werden. Der Benutzer ruft die Prozedur "loesch" auf, wobei er die zu löschenden Einträge durch Matchcodes beim Aufruf spezifiziert:

"Syntax"

loesch Pfad1 (...)

Beispiel: loesch inf* /usr1/klaus/prae* /usr/gast/ddd

Für jeden Eintrag soll abgefragt werden, ob dieser auch wirklich gelöscht werden soll.

Strukturen

Wie geht man nun vor? Zunächst sollte man sich über eine ökonomische Gestaltung Gedanken machen. Man prüft etwa, welche der in Frage kommenden Ablaufstrukturen ("if", "for", "while") für die Problemlösung zweckmäßig ist. Für das Bearbeiten vieler Einträge bietet sich die "for"-Schleife an, während für die Abfragen die "while"-Schleife geeigneter erscheint. Beginnen wir mit der "for"-Schleife zur Abarbeitung der Eintragsnamen, die Abfrage- und Löschmodul umschließen muß (bei komplexeren Prozeduren sind Hilfsmittel wie Nassi-Shneiderman-Diagramme zu empfehlen!):

```
for EINTRAG in $*
# Für sämtliche beim Aufruf angegebenen Parameter:
do
   # Hier sollte nun das noch zu entwickelnde Abfragemodul
   # stehen: "Soll der Eintrag xyz gelöscht werden?"
   # Danach wird das Löschmodul entwickelt und eingebaut.
done
```

Einträge abarbeiten

Abfragemodul:

```
while [ x$antwort != xj -a x$antwort != xn ]
do
   echo "
      Soll der Eintrag $EINTRAG gelöscht werden? (j/n) \c"
   # Der echo-Befehl geht über mehrere Zeilen!
   read antwort
done
```

Einträge abfragen

Löschmodul:

```
if [ x$antwort = xj ]
then
   echo "Der Eintrag \"$EINTRAG\" wird gelöscht!"
   # Die beiden "\" verhindern, daß " interpretiert wird.
   rm -r $EINTRAG
   # Option "-r": Es könnten auch DVZ angegeben werden!
   # Diese werden dann rekursiv gelöscht.
else
   echo "Der Eintrag \"$EINTRAG\" wird nicht gelöscht!"
fi
```

Einträge löschen

Die Prozedur sieht also insgesamt wie folgt aus:

Ergebnis

```
for EINTRAG in $*
do

antwort=
# Der vorherige  Inhalt der  Variablen "$antwort"  muß ge-
# löscht werden,  sonst wird die folgende "while"-Schleife
# u. U. irrtümlicherweise nicht ausgeführt.
while [ x$antwort != xj -a x$antwort != xn ]
do
   echo "
     Soll der Eintrag $EINTRAG gelöscht werden? (j/n) \c"
   read antwort
done

if [ x$antwort = xj ]
then
   echo "Der Eintrag \"$EINTRAG\" wird gelöscht!"
   rm -r $EINTRAG
else
   echo "Der Eintrag \"$EINTRAG\" wird nicht gelöscht!"
fi

done
# Ende der "for"-Schleife.
```

4.6 Beispielprozedur: SQL-Anweisungen in der Shell

Die folgende (für Anfänger nicht einfach zu verstehende) Prozedur zeigt, wie die wichtigsten Mittel zur Erstellung von Shell-Prozeduren ökonomisch verwendet werden. Sie stellt beispielhaft die Integration von INFORMIX-SQL-Anweisungen in eine Shell-Prozedur dar: Sie können nicht einfach in der Shell "select" oder ähnliche SQL-Anweisungen eingeben, da diese nur von INFORMIX verstanden werden.

INFORMIX

Die Prozedur basiert auf der INFORMIX-Datenbank "firma" (im Dateiverzeichnis "Anwendung" in den Beispieldaten). In der Tabelle "auftrag" sind Aufträge und zugehörige Kundennummern gespeichert. Der Benutzer wird zunächst nach einer Kundennummer gefragt. Die Prozedur gibt dann aus, wieviele in der Auftrag-Tabelle enthaltene Aufträge auf den Kunden lauten.

```
KDNR=1000
# Deklaration der Variablen "KDNR"

while [ x$KDNR != x ]
# Die folgenden Kommandos werden ausgeführt, solange die
# Variable $KDNR mit einem Wert besetzt ist
do

echo "Bitte geben Sie die Kundennummer ein: \c"
read KDNR
# Einlesen der Variable KDNR

echo "select count(*) from auftrag
where kd_nr = \"$KDNR\" " > /tmp/$$.sql

# Das  echo-Kommando geht  über 2 Zeilen und erzeugt einen
# SQL-Befehl, der  in der Datei /tmp/$$.sql abgelegt wird.
# $$ ist  die aktuelle Prozeßnummer; da sie im System ein-
# zigartig ist,  wird verhindert,  daß eine Datei gleichen
# Namens schon  existiert oder während des Ablaufs von ei-
# ner anderen Kennung neu angelegt wird (was nicht unwahr-
# scheinlich ist,  wenn die Prozedur Teil eines von mehre-
# ren Kennungen benutzten Programmsystems ist).
# Dabei  wird die  Variable  "$KDNR"  durch  ihren  vorher
# abgefragten  Wert ersetzt  (Variablensubstitution  durch
# Apostrophierung mit "). Die Hochkommata vor der Variable
# müssen mit \ entwertet werden, damit der Interpreter das
# echo-Kommando nicht als abgeschlossen betrachtet.
```

```
echo `isql firma /tmp/$$.sql ¦ \
     sed -e '1,4d' -e '1,$s/^ *//g'` \
"Auftraege laufen ueber die Kundennummer $KDNR."
# Wieder geht der echo-Befehl über mehrere Zeilen.
# In dieser Kommandofolge sind mehrere Funktionen ent-
# halten:
# * echo ...   = Öffnendes echo mit erstem Antwortsatzteil
# * isql ...   = Ausführung der vorher erzeugten SQL-Datei
# * sed  ...   = Löschen der ersten 4 Zeilen der Ausgabe
#                von isql und der führenden Leerzeichen
#                mit Hilfe des Stream-Editors sed (z. B.
#                "   2" zu "2")
# * Auftr...   = Abschluß des echos mit letztem Satzteil
# Die Kommandos isql und sed sind durch eine Pipeline mit-
# einander verbunden (d. h., die Ausgabe von isql wird an
# sed weitergegeben) und in Substitutionsapostrophe (Gra-
# vis) eingeschlossen, so daß die Ausgabe der beiden Kom-
# mandos vom echo-Befehl ausgegeben wird.

done
```

5. Konkrete Problemlösungen für den laufenden Betrieb

5.1 Vorbemerkungen

Die Shell anwenden!

Wie können Sie die bisher erworbenen Kenntnisse bei konkreten Problemen im laufenden Betrieb anwenden? Man kann bei der Mehrzahl der Fälle leider keine generellen Vorgehensanleitungen festlegen, die ein festes Orientierungsschema bieten. Erfahrungsgemäß lernen Sie am besten im laufenden Betrieb, indem Sie sich die notwendigen Kenntnisse aus (Hand-)Büchern zusammensuchen und - oft auf dem Wege von "Versuch und Irrtum" - mögliche Alternativen testen. Im vorliegenden Abschnitt werden beispielhaft nützliche Problemlösungen vorgestellt, für die nach unserer Erfahrung im laufenden Betrieb ein Bedarf entstehen kann. Gleichzeitig illustrieren sie die Denkweise bei der Lösung von Problemen in der Shell.

Im folgenden wird auch deutlich, daß es mit steigender Komplexität der Problemstellung erforderlich wird, in den Handbüchern nachzuschlagen. Die bisher erläuterten Kommandos und Anwendungen stellen nur einen Einstieg in die Shell dar.

Die in den folgenden Abschnitten vorkommenden Prozeduren sind auf der dem Buch beigefügten Beispieldiskette zu finden. Wie die Daten der Beispieldiskette gegliedert sind und wie Sie die Beispieldaten auf Ihren Rechner einspielen, erfahren Sie in Abschnitt 6.1.

5.2 Kopieren von Disketten

Das Kopieren von Disketten erscheint zunächst ein sehr einfacher Vorgang zu sein: Man liest die Diskette mit Hilfe des "tar"-Befehls ein und beschreibt mit dem Disketteninhalt eine andere Diskette. Diese Vorgehensweise birgt jedoch zwei grundlegende Probleme:

Diskette vollständig kopieren

- Sind die Daten auf der Diskette mit absoluten Pfadnamen gesichert, so lassen sie sich nicht ohne weiteres einlesen, insbesondere dann nicht, wenn die Quelldiskette von einer Partition beschrieben wurde, die auf dem eigenen Rechner gar nicht existiert.
- Der "tar"-Befehl liest (bzw. schreibt) nur von einem (bzw. auf einen) Bereich der Diskette.

Um diese Probleme zu lösen, muß man wissen, über welche "Device"-Namen das Diskettenlaufwerk angesprochen wird. Hier eine Auflistung der ("fl̲oppy"-)Gerätedateien:

Diskettenbereiche

/dev/fl0: Labelbereich der Diskette
/dev/fl1: Bootbereich der Diskette
/dev/fl2: Datenbereich der Diskette

Auf dem Labelbereich der Diskette werden der Name des Eigentümers und eine Versionsnummer (für Softwareprodukte relevant) eingetragen. Im Bootbereich stehen ausführbare Programme wie z. B. die Prozedur "install", die eine automatische Installation der Software von der Diskette ermöglicht. Der Datenbereich enthält die eigentlichen Daten der Diskette, die z. B. mit dem "tar"-Befehl bearbeitet werden können.

Die Lösung des Problems führt über den Weg der Standard-Ausgabe-Umleitung. Die Daten können über ihre Gerätedateien direkt angesprochen werden. Um die Inhalte der drei Bereiche in Dateien auf der Festplatte umzuleiten (hier "label", "boot" und "daten" genannt), geben Sie die folgenden Befehle ein (legen Sie vorher die Quelldiskette ein!):

Gerätedateien

Quelldiskette

```
cat /dev/fl0 > label
cat /dev/fl1 > boot
cat /dev/fl2 > daten
```

Um diese Daten nun auf die Zieldiskette zu speichern, kehrt man die obigen Argumente um (legen Sie vorher die Zieldiskette in das Laufwerk!):

Zieldiskette

```
cat label > /dev/fl0
cat boot  > /dev/fl1
cat daten > /dev/fl2
```

Auf der Beispiel-Diskette finden Sie die Shell-Prozedur "copyfl", die diesen Mechanismus automatisiert.

5.3 Organisation des Druckens auf fernen Rechnern

Rechner werden immer häufiger in Local Area Networks (LAN) vernetzt. Vorteilhaft ist dabei vor allem die Möglichkeit, periphere Geräte und Kapazitäten der anderen Rechner im Netz zu nutzen. Zum Beispiel ist das Einlesen von Disketten auf Rechnern, die in einem Rechenzentrum stehen, in der Regel nicht möglich, weil der Zugang aus Sicherheitsgründen beschränkt ist oder auch nur, weil der Rechner recht weit entfernt steht. Man bindet dann in den einzelnen Abteilungen "Disketten- und Druck-Server" in das lokale Netz ein, von denen aus Disketten auf den Rechner im Rechenzentrum eingelesen werden können. Ebenso kann an einen solchen Server ein besonderer Drucker angeschlossen werden.

Disketten-server

Netz-Drucken

Zum Drucken auf einem fernen Drucker sollte das "lpr"-Kommando auf dem fernen System mit Hilfe des "rsh"-Befehls angesprochen werden. Man bildet eine Pipeline, in die die Eingabe für den Druckbefehl auf dem fernen Rechner geschickt wird. Hierzu benötigt man eine Kennung im fernen System. **Jede** aufrufende Kennung muß auf dem fernen Rechner in der Datei "**.rhosts**" unter dem Stamm-Dateiverzeichnis der Kennung eingetragen sein, die sozusagen als "Träger" für den "rsh"-Befehl verwendet wird. Im folgenden soll die "Träger"-Kennung auf dem fernen Rechner "netzprn" heißen. Somit könnte der Pfad z. B. "/usr1/netzprn/.rhosts" lauten. Hier die allgemeine Syntax des Eintrags:

.rhosts

Rechnername Kennung
(...)

Die Datei könnte z. B. wie folgt aussehen:

```
MX300-1  admin
mx-pc  maurer
MX300-1  schwickert
MX300-3  vertrieb
MX300-2  zentrale
```

Das Drucken auf einem fernen Drucker kann mit Hilfe einer von jedem Benutzer aufrufbaren Shell-Prozedur "rlpr" realisiert werden. Sie kann unter dem Pfad "/usr/bin" mit Ausführberechtigung für alle abgelegt werden und mit folgender Syntax aufzurufen sein:

rlpr-Syntax

rlpr Host (Datei1) (...) (Schalter)

VORSICHT:
Die exakte "lpr"-Syntax kann nicht verwendet werden: z. B. würde "rlpr" aus "rlpr datei1 -pb2 datei2 -pb3" den folgenden (fehlerhaften) "lpr"-Befehl zusammenstellen:

lpr -pb2 -pb3 datei1 datei2

Die Prozedur faßt also einfach alle Schalter sowie Dateien nacheinander zusammen und übergibt Sie dem "rsh"- bzw. dem "lpr"-Befehl. Um die Dateien des Beispiels wie intendiert auszudrucken, müßte man "rlpr" zweimal aufrufen.

Die Prozedur "rlpr" könnte wie folgt aussehen:

Prozedur "rlpr"

```
# Prozedur: rlpr: lpr fuer Remote-Drucken aus der Shell

SCH=
DAT=
# Deklaration der Variablen $SCH und $DAT;
# $SCH enthält Schalter, $DAT speichert Dateinamen.

if [ x$1 != x ]
then
  RH=$1
  # Der erste Stellungsparameter $1 (hinter dem Namen der
  # Prozedur beim Aufruf) wird als Wert der Variablen
  # $RH zugewiesen: Name des fernen Rechners (Remote Host)
  shift
else
  echo "Fehler! Kein Remote-Host angegeben!"
# Wenn kein Argument hinter dem Aufruf angegeben wird,
# wird ein Syntaxfehler angenommen (es wurde kein ferner
# Rechner angegeben).
  exit 1
fi

while [ x$1 != x ]
# Die Schleife wird ausgeführt, solange Stellungsparameter
# (beim Aufruf der Prozedur angegeben) vorhanden sind.
do
  if [ -f "$1" ]
  # Wenn der aktuelle Stellungsparameter eine Datei ist:
  then
    DAT="$DAT $1"
    # Die Variable $DAT wird um den aktuellen Stellungs-
    # parameter erweitert (z. B.: "dat1" zu "dat1 dat2").
```

```
  else
    SCH="$SCH $1"
    # Wenn der Parameter keine Datei ist, wird er als ein
    # Schalter für den "lpr"-Befehl interpretiert.
  fi
    shift
    # Verschieben zum nächsten Stellungsparameter.
done

for DRUCKDAT in $DAT
# Für jeden in $DAT gespeicherten Dateinamen wird die
# folgende Kommandozeile einmal ausgeführt (jede
# Komponente der Sammelvariable $DAT wird pro Schleifen-
# durchlauf der Variable $DRUCKDAT zugeordnet)
do
  cat "$DRUCKDAT" ¦ rsh "$RH" -l netzprn lpr "$SCH"
  # Aufbereitung zum Drucken; $RH ist der Remote-Host.
done

if [ x$DAT = x ]
# Wenn keine Dateien zum Drucken angegeben wurden
# ($DAT ist leer); auch wird bei der Verwendung von
# rlpr in einer Pipeline die obige Schleife nicht
# ausgeführt
then
  rsh "$RH" -l netzprn lpr "$SCH"
fi
```

Beispiele für den Aufruf des selbstentwickelten Befehls: Anwendung

Drucken der Dateien "brief" und "beschwerde" auf dem Rechner "MX3" in Schriftbreite 12 und nationalem Zeichensatz auf der Druckergruppe "LASER":

```
rlpr MX3 brief beschwerde -pb2 -dt -dru=LASER
```

Man kann "rlpr" auch in einer Pipeline verwenden, z. B. eingebaut in einen Druckeraufruf des Textsystems HIT (Pfad der Druckeraufrufe: "/usr/att/usr/lib/hit/printer"):

rlpr in HIT

```
# 9001: Drucker 9001
trap "" 1 2 3 15

filter -d pr9001 -l 70 -p -S $2 -r $3 $1 ¦ \
rlpr MXX -dru=G11 -nc=$4 -tl=$5 -pr=$6
# Hier wurde nur "lpr" durch "rlpr" ersetzt und der Name
# des fernen Rechners ("MXX") sowie die Druckergruppe
# (auf dem fernen Rechner) eingetragen.
```

5.4 Empfehlungen für die Datensicherung

Festplatten sind wie andere mechanische Teile auch von Abnutzungen und anderen Fehlern betroffen. Die schlimmste Konsequenz einer defekten Festplatte ist der Verlust unwiederbringlicher Daten. Datenverluste können auch bei "Systemabstürzen" auftreten. Deshalb ist die Datensicherung eine elementare Aufgabe des Systemverwalters, die mindestens wöchentlich durchgeführt werden sollte. Man kann damit den Schaden von Datenverlusten auf ein kalkulierbares Maß beschränken.

Datensicherung

Die folgenden Ausführungen beschränken sich auf die Verwendung von Magnetbandkassetten (MBK), da die Sinix-Geräte standardmäßig mit Magnetbandkassettenlaufwerken ausgerüstet sind, mit denen die Datensicherung relativ schnell und zuverlässig durchgeführt werden kann. Wenn Sie z. B. ein Magnetbandlaufwerk anschließen, sollten Sie sich informieren, welche Gerätedatei dann anstelle von "/dev/rts0" angegeben werden muß.

MBK-Sicherung

In Sinix bieten sich prinzipiell drei Alternativen an:

tar

1. Datensicherung mit dem Befehl "tar"

Syntax des Befehls zum Sichern auf MBK:
tar cvf /dev/rts0 Pfade

Syntax des Befehls zum Wiedereinlesen einer Sicherung:
tar xvf /dev/rts0 (Pfade)

Der "tar"-Befehl ist einfach zu handhaben und zumindest beim Sichern relativ schnell. Man kann bequem selektive Pfade vom Sicherungsband wieder einlesen (ein Inhaltsverzeichnis der Kassette erhält man mit dem Schalter "t" anstelle von "c"), und das System muß sich nicht im Ein-Benutzer-Betrieb befinden (wie bei "dump"/"restore").

2. Datensicherung mit dem Kommando "cpio"

cpio

Syntax des Befehls zum Sichern auf MBK:
find Pfade -print ¦ cpio -o > /dev/rts0

Es ist auch möglich, die zu sichernden Daten mit "> Datei" in eine Datei umzuleiten und dann mit "tar" zu speichern. Erstens ist dies jedoch zu kapazitätsintensiv (die Daten müssen erst auf die Platte geschrieben und dann wieder gelöscht werden), und zweitens kann der Vorteil des direkten selektiven Zurückschreibens dann nicht genutzt werden.

Wiedereinspielen dieser Daten mit dem Befehl:
cpio -id(u) (Pfad1) (...) < /dev/rts0

Sie können nun selektiv zurückspeichern, indem Sie spezielle Pfade angeben.

Sie können besonders "cpio" und "tar" in Form einer Shell-Prozedur benutzerfreundlich ausgestalten. Die Shell bietet aber noch erheblich mehr Ausgestaltungsformen. Zum Beispiel können Sie über die Datei "/usr/.lib/crontab" (vgl. Abschnitt 5.5 "Automatisches Ausschalten des Systems") eine periodische Datensicherung "erzwingen", indem auf sämtlichen Bildschirmen eine Aufforderung zur Datensicherung angezeigt wird.

3. Datensicherung mit "/etc/dump" und "/etc/restore"

dump/restore

Dieses Konzept wird allgemein empfohlen. Die beiden Kommandos beanspruchen das Laufwerk nicht so sehr wie z. B. "tar", denn mit "dump" kann in Sicherungsstufen gearbeitet werden: In Folgesicherungen wird "selektiv" nur das gesichert, was seit der letzten Sicherung verändert wurde. Nachteilig ist die schlechte Transparenz der Kommandos; vor allem "restore" ist kompliziert zu handhaben. Es ist z. B. prinzipiell möglich, mit "restore" Daten selektiv von der MBK zurückzuholen, die dazu notwendigen Kommando-Variationen sind jedoch nicht ohne weiteres verständlich.

Beispiele finden Sie in Abschnitt 3.6 "Kommandos für die Systemverwaltung". Die weitergehende Verwendung von "restore" kann man im Handbuch "Betriebssystem Sinix V5.xx - Systemverwaltung" nachlesen.

5.5 Automatisches Ausschalten des Systems

Wenn Sie das System nicht jedesmal am Ende der Woche manuell abschalten wollen, können Sie die Möglichkeit der automatischen Abschaltung durch Sinix nutzen. Manuell kann der Systemverwalter das System mit dem Befehl "/etc/shutdown" oder auch softwaregesteuert mit der Kennung "shutdown" (ruft eben dieses Kommando auf) ausschalten.
Aus Gründen der Datensicherheit, des Energiesparens oder der Bequemlichkeit kann eine automatische Abschaltung z. B. am Ende jeder Woche erwünscht sein. Diese Aufgabe kann am besten mit Hilfe der Systemdatei **"crontab"** gelöst werden, die zu genau festgelegten Zeitpunkten bestimmte Kommandos ausführt. Sie ist unter dem Pfad "/usr/.lib/crontab" zu finden.

crontab

In diese Datei trägt man den Befehl "/etc/shutdown" ein. Es empfiehlt sich, nicht den Schalter "-h now" zu verwenden, da das System in diesem Fall sofort heruntergefahren würde. Etwaige noch arbeitende Benutzer hätten dann keine Zeit mehr, ihre Daten (z. B. im Textsystem HIT) zu sichern. Deshalb stößt man die "shutdown"-Prozedur mit einer Verzögerung von z. B. 5 Minuten an. Während dieser Zeit bekommen sämtliche noch aktive Benutzer jede Minute die Mitteilung, daß das System heruntergefahren wird - mit Angabe der verbleibenden Zeit.

Hier soll es genügen, die Syntax für die Systemtabelle "crontab" anhand einer Abschaltung zu erklären, die freitags um 18 Uhr angestoßen wird. Eine ausführlichere Syntax-Erklärung können Sie im Handbuch "Betriebssystem Sinix V5.xx - Systemverwaltung" nachlesen. Die Einträge werden zeilenweise vorgenommen und bestehen aus sechs Komponenten, wie die folgende Beispielzeile zeigt:

crontab-Syntax

```
0    18   *    *    5    /etc/shutdown -h '+5'
a)   b)   c)   d)   e)   f)
```

a) Minute: In welcher Minute einer Stunde soll das Programm ausgeführt werden (0 bis 59)?

b) Stunde: Stunde des Aufrufs (0 - 23)

c) Tag/Monat: Tag (im Monat) des Aufrufs (1 - 31)

d) Monat: Monat des Aufrufs (1 - 12)

e) Tag/Woche: Wochentag des Aufrufs (1 - 7, Montag = 1)

f) Programm: Name des auszuführenden Kommandos

Dabei können mit "*" alle bzw. beliebige Werte, mit "Zahl,Zahl(,...)" (z. B. "1,5,10") eine Liste von Werten und mit "Zahl-Zahl" (z. B. "1-5") ein Bereich von Werten spezifiziert werden.

Damit hat die obige Beispielzeile folgende Wirkung: Zu Beginn der ersten Minute ("0"), in der 18. Stunde (18 Uhr, "18"), an einem beliebigen Tag ("*") in einem beliebigen Monat ("*") und an einem Freitag ("5") wird das Kommando "/etc/shutdown -h '+5'" ausgeführt (immer unter der Berechtigung von "admin").

5.6 Kopieren einer INFORMIX-Datenbank

INFORMIX-DVZ kopieren

Im Lehrbetrieb tritt häufig das Problem auf, daß Kursteilnehmer auf eine INFORMIX-Beispieldatenbank oder einen SIPLAN-Beispielordner zugreifen sollen. Das gleiche Problem kann sich im Unternehmen stellen, wenn z. B. eine Datenbank anderen Benutzern zur Verfügung gestellt werden soll. Da bei den häufig benötigten INFORMIX-Datenbanken besondere Probleme entstehen, soll das Kopieren einer INFORMIX-Datenbank hier beispielhaft vorgestellt werden.

Die Datenbank ist ein Dateiverzeichnis mit dem Datenbanknamen und dem Suffix ".dbs" (z. B. "vertrieb.dbs"). Dieses Dateiverzeichnis enthält Systemdateien, die die eigentlichen Daten enthalten, z. B. INFORMIX-interne Zugriffsdaten und Index-Dateien. Die Programme (Masken-/Listenprogramme und SQL-Befehle) legt INFORMIX standardmäßig im gleichen Dateiverzeichnis ab, in der die Datenbank zu finden ist. Wenn Sie nur die Datenbank kopieren wollen (ohne die Programme), müssen Sie den Pfad der Datenbank angeben, andernfalls kann der gesamte Ordner, der Datenbank und Programme enthält, rekursiv kopiert werden. Einzelne Tabellen können nicht kopiert werden.

Die Besonderheit bei INFORMIX-Daten sind die "dualen" Zugriffsberechtigungen: Sowohl die Berechtigungen in der Shell (Schutzbit-Einstellungen) als auch die INFORMIX-intern vergebenen Zugriffe müssen stimmen, damit ein Benutzer mit den kopierten Daten auch arbeiten kann.

Gehen Sie wie folgt vor, wenn Sie eine Datenbank kopieren wollen:

Lokalisierung der Daten

1. Finden Sie die absoluten Pfadnamen der Daten heraus. Mit

   ls Pfade ¦ pg   oder   find Pfad Bedingung Aktion

   können Sie prüfen, ob die Dateien existieren.

Shell-Zugriffe prüfen

2. Besteht auf Shell-Ebene Leseberechtigung auf die zu kopierenden Dateiverzeichnisse und Dateien bzw. Schreibzugriff auf das Ziel-Dateiverzeichnis (in dem die Daten abgelegt werden sollen)? Prüfen Sie die Zugriffseinstellungen mit dem folgenden Kommando:

   ls -l Pfade ¦ pg

   Wenn Sie als Systemverwalter arbeiten, müssen Sie nicht auf Lese- und Schreibberechtigung achten.

Shell-Zugriffe vergeben

3. Falls Sie in der Shell noch kein Leserecht auf die Original-Datenbank besitzen, muß der Eigentümer der Daten vorher den folgenden Befehl absetzen (der Systemverwalter kann auch ohne Leserecht kopieren):

   chmod a+r Pfade

   Hilfreich sind hier Matchcodes: chmod a+r * */* */*/*

INFORMIX-Rechte prüfen

4. Sind auch in INFORMIX die korrekten Zugriffsrechte gesetzt? Prüfen Sie die INFORMIX-Zugriffseinstellungen, indem Sie in der Shell zu dem Dateiverzeichnis wechseln, der die Datenbank enthält, und den Shell-Befehl "isql Datenbankname" eingeben. Ein Beispiel:

   ```
   cd /usr1/inform/DBordner
   isql kundenDB
   ```

   Innerhalb der Funktion "SQL-Dialog" des INFORMIX-Menüs können Sie sich im Menüpunkt "Info" über bestehende INFORMIX-Zugriffsrechte informieren.

5. Die Vergabe der gewünschten INFORMIX-Zugriffsrechte ist Sache des Eigentümers der Datenbank. Den INFORMIX-Zugriffsschutz kann nicht einmal der Systemverwalter umgehen. Der Eigentümer muß im INFORMIX-Menü "SQL-Dialog" den Menüpunkt "Neu" auswählen und z. B. den folgenden SQL-Befehl eingeben (der Befehl wird ausgeführt, wenn Sie auf die Taste <START> drücken):

INFORMIX-Rechte vergeben

```
grant dba to gast
```

Hier wurde dem Benutzer "gast" (der die Daten kopieren möchte) praktisch alles erlaubt: Er kann nun Tabellen anlegen und löschen sowie Daten manipulieren. Um eine Datenbank mit Grundfunktionen bearbeiten zu können, muß die betreffende Kennung mindestens "CONNECT"-Erlaubnis besitzen ("grant connect to Kennung"). Weitere Zugriffsfunktionen finden Sie im Handbuch "INFORMIX Nachschlagen" unter dem RDSQL-Befehl "GRANT".

6. Nun können Sie in der Shell die Daten kopieren:

Kopieren in der Shell

Im Siemens-Universum:

```
copy -r Quell-DVZ Ziel-DVZ
```

Im AT&T-Universum:

```
mkdir Ziel-DVZ ; cd Quell-DVZ ; cd .. ;
find Quell-DVZ -print ¦ cpio -pdu Ziel-DVZ
```

"Quell-DVZ" ist entweder der Ordner, in dem sich das Datenbank-DVZ ("Datenbankname.dbs") befindet, oder der Datenbank-Katalog selbst. Die Erzeugung des Ziel-DVZ (Pfad, unter dem die Kopie stehen soll) mit "mkdir" ist nicht notwendig, wenn es schon existiert.

Zugriffe (Systemverwalter)

7. Wenn Sie als Systemverwalter Daten zu einer anderen Kennung kopiert haben, müssen Sie noch zusätzlich die Zugriffseinstellungen richtigstellen, damit die Zielkennung mit den Daten arbeiten kann:

chown Kennung Pfade ; chmod Zugriffe Pfade

Mit "chown" machen Sie die betreffende Kennung zum Eigentümer der Daten, mit "chmod" können Sie Lese-, Schreib- und Ausführberechtigung vergeben.

Bei Pfadangaben spart die Verwendung von Matchcodes Arbeit. Beispiel: chown gast * */* */*/*

Hinweis: Um eine Datenbank, die in der Shell einer anderen Kennung gehört, bearbeiten zu können, muß der betreffende Benutzer Schreiberlaubnis ("rw") besitzen. Wenn Sie also die Daten als Systemverwalter zu einer anderen Kennung kopiert haben, müssen Sie entweder die betreffende Kennung zum Eigentümer des Datenbank-Dateiverzeichnisses (und der darunterliegenden Dateien) machen (Kommando "chown") oder für die Kennung Schreibrecht vergeben (Kommando "chmod").

Prozedur

Wie man mit Hilfe einer Shell-Prozedur diesen Vorgang automatisiert, können Sie mit Hilfe der Prozedur "bspkop.scr" auf der Beispieldiskette nachvollziehen.

Beispiele und Erläuterungen

Beispiel

Die Benutzerin "julia" möchte den Ordner "DB" mit der Datenbank "kurse" aus der Kennung "verwalt" kopieren. "Quell-DVZ" ist also "/usr1/verwalt/DB" und "Ziel-DVZ" "$HOME/julia":

1./2. Prüfen der Shell-Zugriffsrechte (durch "julia"):

```
ls -ld /usr1/verwalt/DB
# Zugriffe auf den Ordner auflisten
ls -lR /usr1/verwalt/DB ¦ pg
# Zugriffe auf den Inhalt des Ordners "DB" auflisten
```

<u>Mögliches Ergebnis:</u> Schutzbiteinstellung "rwx------" für alle Dateien und Dateiverzeichnisse (nur der Eigentümer hat alle Rechte, es müssen also Zugriffe durch den Eigentümer, in diesem Fall "verwalt", gesetzt werden).

3. Setzen der Zugriffsrechte (Leserecht für alle) durch die Kennung "verwalt" in der Shell:

```
cd /usr1/verwalt
chmod a+rx DB DB/* DB/*/*
```

4. Prüfen der INFORMIX-Zugriffsrechte:

Die Benutzerin "julia" prüft ab, ob sie die Datenbank aus dem Menü heraus bearbeiten kann, indem sie sie mit dem Befehl "isql kurse" aus der Shell aufruft. Wenn sie die Fehlermeldung "... keine CONNECT-Erlaubnis ..." (oder ähnlich) erhält, teilt sie der Kennung "verwalt" mit, daß sie Zugriff auf die Datenbank "kurs" erteilt haben möchte.

5. INFORMIX-Zugriffe setzen (durch die Kennung "verwalt"):

```
cd /usr1/verwalt
cd DB
isql kurse zugriff.sql
# Somit sind die INFORMIX-Zugriffe gesetzt worden;
# Die Datei "zugriff.sql" hat folgenden Inhalt:
#       grant dba to julia
```

Wenn "verwalt" nicht möchte, daß die ursprüngliche, in seinem Ordner DB befindliche Datenbank versehentlich oder absichtlich von "julia" verändert wird (was nur geschehen kann, wenn "julia" neben den INFORMIX-Rechten auch Schreiberlaubnis in der Shell hat), erstellt er eine Arbeitskopie in einem anderen Ordner, die dann kopiert und bearbeitet werden kann.

6. Kopieren der Datenbank (unter "julia"):

```
mkdir $HOME/DB
# Erzeugen des Ordners "DB" im eigenen Stamm-DVZ
cd /usr1/verwalt
find DB -print ¦ cpio -pdu $HOME
# Rekursives Kopieren des gesamten Ordners
```

Im Siemens-Universum geht dies mit dem folgenden Befehl:

```
copy -r /usr1/verwalt/DB $HOME/DB
```

5.7 Protokollieren der Nutzungszeiten: Accounting

Accounting

Aufgabe des Accounting ist es, für jede einzelne Kennung im System die Nutzungszeiten zu protokollieren. Diese Zeiten sind Kennzahlen für den Verbrauch der Systemressourcen (Inanspruchnahme des Zentralprozessors (CPU), des Hauptspeichers, der Festplatte, der E/A-Prozessoren). Wichtig ist die Feststellung des Verbrauchs pro Kennung vor allem bei betrieblich genutzten Mehrplatzsystemen, die von mehreren Abteilungen in Anspruch genommen werden. Will die Kostenrechnungsstelle die Kosten gemäß den Nutzungszeiten der einzelnen Kennungen auf die verschiedenen Kostenstellen umlegen, so muß man wissen, wer wieviel Systemzeit verbraucht hat. - Hier wird das Accounting für Sinix V5.2x beschrieben.

Das Kommando zum Ein- und Ausschalten des Accountings (standardmäßig nur durch den Systemverwalter aufzurufen) lautet wie folgt:

Ein- und Ausschalten: accton

```
/usr/etc/accton (Datei)
```

Wenn Sie dieses Kommando mit Angabe eines Dateinamens absetzen, schalten Sie das Accounting ein. Die betreffende Datei muß bereits existieren. Wenn Sie den Befehl danach ohne Argument aufrufen, schalten Sie das Accounting wieder aus.

Durch den Aufruf des Kommandos mit Angabe eines Dateinamens können Sie auch eine andere Protokolldatei wählen als die standardmäßige (schon vorhandene) Datei

Protokolldatei

"/usr/adm/acct", wenn ein anderer Dateiname als beim Einschalten angegeben wurde. Die Datei liegt nicht in ASCII-Klartext vor (ist z. B. nicht mit "pg" zu lesen!) und wird ständig aktualisiert.

Das folgende Kommando wertet Protokolldateien aus:

```
/usr/etc/sa -m (Datei1) (...)
```

Auswerten: sa -m

Wenn "Datei" nicht angegeben ist, wird standardmäßig die Datei "/usr/adm/acct" ausgewertet.

Beispielausgabe von "/usr/etc/sa -m":

Beispiel

```
shutdown   23   0.07cpu   137tio    77k*sec
verwalt    25   0.33cpu   458tio   680k*sec
```

Erläuterung:

1. Spalte: Angabe der Kennung
2. Spalte: Anzahl der Prozeß-Aufrufe
3. Spalte: User- und System-Zeit der Benutzung des Prozessors (in Minuten)
4. Spalte: Anzahl der (Terminal-)Input/Output-Operationen
5. Spalte: CPU-Zeit der durchschnittlichen Hauptspeicherbelegung ("kilo-core seconds")

Die Benutzer- und Systemzeiten in der dritten Spalte geben die Nutzungsanteile der einzelnen Kennungen relativ realistisch wieder und können dementsprechend auch z. B. für die Verrechnung von Accounting-Auswertungen mit Hilfe von Kostensätzen verwendet werden.

Im obigen Beispiel werden alle Systemprozesse, die durch Systemkennungen wie z. B. "root", "admin" und "daemon" ausgeführt werden, unter der Kennung "**shutdown**" subsumiert.

VORSICHT: Unter Sinix V5.21 werden die verbrauchten CPU-Zeiten auf der MX500 um den Faktor 100 zu hoch angegeben!

<u>Beispiel einer systematischen Accounting-Auswertung</u>

Ein Beispiel, wie mit der Hilfe einer Shell-Prozedur Accounting-Auswertungen direkt in eine INFORMIX-Datenbank geladen werden können, ist auf der beigelegten Beispiel-Diskette zu finden (Datei "Anwendung/auswert").

Accounting: Anwendungsbeispiel

"Einhängen" des Accounting-Systems in das Betriebssystem

Accounting "einhängen"

Wie bindet man das Accounting-System am besten in die bestehende System-Umgebung ein?

Um das Accounting sofort beim Systemstart zu aktivieren (damit auch wirklich alle Nutzungszeiten erfaßt werden), schreibt man den Befehl zum Einschalten in die Datei "/etc/rc". Diese Shell-Prozedur wird sofort beim Einschalten des Systems ausgeführt und lädt alle notwendigen Systemprogramme (z. B. Drucker-Treiberprogramme). In Sinix V5.21 sieht die Datei "rc" vor der Veränderung wie folgt aus (Auszug der letzten Zeilen):

/etc/rc

```
                                echo -n standard daemons:
/etc/update &                   echo -n ' update'
/etc/cron &                     echo -n ' cron'
# /usr/etc/accton /usr/adm/acct &   echo -n ' accounting'
/usr/spool/spooler/startup rc & echo -n ' spooler'
exit 0
```

Der Accounting-Befehl wird durch das Kommentarzeichen "#" außer Kraft gesetzt. Wenn Sie die Datei (als Systemverwalter, also unter "root" oder "admin") im CED editieren und das Kommentarzeichen entfernen, wird das Accounting-System bei jedem Systemstart aktiviert.

Monatliches Speichern der Accounting-Daten

Monats-auswertungen

Betriebliche Kosten werden monats- oder quartalsweise abgerechnet. Eine jährliche Accounting-Auswertung wäre also kaum sinnvoll. Offensichtlich ist es in diesem Fall hilfreich, die Daten monatsweise in Dateien zu speichern. Wie kann man dieses Problem lösen?

Protokolldatei

Zunächst ist eine Konvention für Dateinamen festzulegen. Da für jeden Monat eine Datei angelegt werden sollte, bietet sich folgende Namensgebung an: z. B. für den Monat Juli im Jahr 1991 "**acct071991**" ("acct" für Accounting, "07" für Juli und "1991" für das Jahr). Mit dieser Konvention können die Dateinamen in Shell-Prozeduren automatisch erzeugt werden.

Da die Dateien mit den Accounting-Daten im Laufe der Zeit sehr groß werden, ist ihre Verlagerung auf eine kapazitätsmäßig weniger ausgelastete Partition als "/usr" empfehlenswert. Das Dateiverzeichnis kann z. B. "/usr1/ACCT" heißen. (Der Systemverwalter muß dieses DVZ mit dem "mkdir"-Kommando einrichten.) Der Pfad für die Oktober-Auswertung von 1991 wäre dann "/usr1/ACCT/acct101991".

Beim Einschalten des Systems muß geprüft werden, ob die Accounting-Datei für den aktuellen Monat schon existiert. Falls notwendig, ist eine neue anzulegen (automatische Erzeugung des neuen Dateinamens). Außerdem muß verhindert werden, daß Daten in eine falsche Datei eingetragen werden. Wenn das System über das Monatsende hinaus eingeschaltet ist, wird der in "/etc/rc" festgelegte Dateiname weiterverwendet. Um dies zu vermeiden, kann ein entsprechender Eintrag in der Datei "/usr/.lib/crontab" gemacht werden (vgl. Abschnitt 5.5 "Automatisches Ausschalten des Systems"): Dieses Programm prüft unter Systemverwalter-Berechtigung jeden Tag, ob ein neuer Monat beginnt.

crontab

Die in den Dateien "rc" und "crontab" durchzuführenden Aktionen sind wie folgt zusammenzufassen:

Aktionen

- Namen der korrekten Protokolldatei erzeugen;
- Accounting mit der richtigen Protokolldatei einschalten;
- Accounting auf die richtige Protokolldatei umschalten;
- die Datei erzeugen, wenn sie noch nicht existiert.

Es bietet sich also an, eine kurze Shell-Prozedur zu schreiben, die aus "rc" und "crontab" aufzurufen ist. Im folgenden soll diese Prozedur "umschalt.scr" genannt werden (da sie das Accounting auf die richtige Datei ein- bzw. umschaltet). Hier der Aufruf in der Datei "/etc/rc":

att sh /usr/admin/umschalt.scr & echo -n ' accounting' /etc/rc

In die Datei "/usr/.lib/crontab" muß die folgende Zeile eingetragen werden:

0 0 1 * * sh /usr/admin/umschalt.scr crontab

Die Prozedur (hier abgelegt unter "/usr/admin") wird also zu Beginn der ersten Stunde des ersten Tages eines jeden Monats aufgerufen.

Die Shell-Prozedur sieht wie folgt aus:

Shell-Prozedur

```
# umschalt.scr: Aktiviert Accounting mit der richtigen
#               Datei (Universum: att)

DATEI=/usr1/ACCT/acct`date '+%m'`19`date '+%y'`
# Erzeugen des Dateinamens (in Variable $DATEI gemerkt):
# /usr1/ACCT/acct ist die erste Komponente des Namens;
# Erweiterte Syntax des date-Befehls: date '+%Wert',
# wobei der Wert in Akzente eingeschlossen werden muß (').
# Wenn "Wert"  z. B. "y"  ist, werden  die letzten  beiden
# Stellen der Jahreszahl (z. B. "91") ausgegeben.
# `date '+%m'`,  die zweite Komponente des Namens (Monats-
# angabe), wird durch Kommandosubstitution (`) erzeugt;
# 19 ist die dritte Namenskomponente (konstanter Wert);
# `date '+%y'` ist die vierte Komponente (Jahresangabe).

if [ ! -f $DATEI ]
# Wenn die Datei noch nicht existiert ("test"-Befehl)
then
  > $DATEI
  # Anlegen der Datei
fi

/usr/etc/accton $DATEI
# Einschalten des Accountings mit der richtigen Datei
```

Wenn man den Platz reduzieren will, den die großen Protokolldateien auf der Festplatte einnehmen, kann man sie mit dem Kommando "pack" komprimieren.

6. Übungsaufgaben

6.1 Installation und Gliederung der Beispieldaten

Um die Daten der Beispieldiskette auf die Festplatte in Ihr Stammverzeichnis zu schreiben, führen Sie bitte die folgenden Schritte durch: Installation

1. Melden Sie sich beim Betriebssystem unter Ihrer Kennung an (Login). Wechseln Sie in die Shell, falls Ihre Kennung nicht der Klasse "expert" angehört.

2. Legen Sie Diskette in das Laufwerk ein.

3. Geben Sie in der Shell (das Universum ist beliebig) den folgenden Befehl ein (fl2 steht für **floppy 2**):

```
tar xvf /dev/fl2
```

Die Daten auf der Beispieldiskette sind in zwei Dateiverzeichnisse gegliedert: Gliederung

Bsp: Beispieldateien für die Übungsaufgaben;
Anwendung: Ausgewählte Shell-Prozeduren des Kapitels 5 "Konkrete Problemlösungen für den laufenden Betrieb" sowie Prozeduren zur Demonstration.

In den Beispieldaten für die Übungsaufgaben beginnen zur besseren Übersicht die Namen der Dateiverzeichnisse mit Großbuchstaben und Dateinamen mit Kleinbuchstaben. Die Datei **"inhalt"** enthält eine Übersicht der enthaltenen Dateien und Dateiverzeichnisse. Wenn Sie sich dieses Dokument anschauen wollen, geben Sie - nach der Installation der Beispieldaten und gegebenenfalls einem Wechsel ins AT&T-Universum mit dem Befehl "att" - bitte die folgenden Kommandos ein:

```
cd Bsp
pg inhalt
```

Datei Bsp/inhalt

Die Shell-Prozeduren im Dateiverzeichnis "Anwendung" enthalten nähere Erläuterungen in Form von Kommentarzeilen, die mit dem Zeichen "#" eingeleitet werden. Die ebenfalls dort abgelegte Datei "inhalt" enthält kurze Beschreibungen der enthaltenen Dateien (und Dateiverzeichnisse). Auch dieses Dokument können Sie mit den Kommandos

Datei Anwendung/inhalt

```
cd Anwendung
pg inhalt
```

am Bildschirm ansehen.

Ausdrucken

Sie können die für Sie interessanten Dateien mit dem folgenden Befehl ausdrucken (die Druckausgabe erfolgt dann auf dem Standarddrucker):

```
lpr -pb2 -int Datei1 (...)
```

6.2 Aufgaben

Zum Kennenlernen von Sinix ist es sicher nicht ratsam, die als Nachschlagewerk gedachte Kommandoauflistung in Kapitel 3 sequentiell durchzuarbeiten. Wir stellen Ihnen daher im folgenden eine Reihe von Übungsaufgaben, an denen Sie wachsen können - mit vielen Hilfestellungen am Anfang und anspruchsvolleren Aufgaben am Ende. Auch später werden Sie feststellen, daß die Lösung konkreter Problemstellungen im laufenden Betrieb besser ist als jedes Schulungswerk. Versuchen Sie also, wann immer es möglich ist, die gelernten Kommandos anzuwenden.

Bevor Sie mit den Aufgaben beginnen, bitten wir Sie, die beiden Grundlagenkapitel 1 und 2 zu lesen. Von jeglicher Sachkenntnis völlig ungetrübt, macht das Arbeiten am Rechner zwar manchen Menschen besonders viel Spaß, führt aber bei den meisten schnell zu akutem Lustverlust (Frust). Besonders Ungeduldigen sei gesagt, daß für den ersten Abschnitt der Übungen die Lektüre der Abschnitte 1.1, 1.2 und 2.1 genügt - ab dem zweiten Abschnitt werden jedoch die Kenntnisse der ersten zwei Kapitel vorausgesetzt.

Lassen Sie sich bei der Lösung der Aufgaben von der Kurzauflistung der Kommandos am Ende dieses Buches "inspirieren": Suchen Sie dort nach Kommandos, die Ihnen für eine Problemstellung geeignet erscheinen, und schlagen Sie unter diesen Kommandos nach. Wenn Sie so nicht weiterkommen, finden Sie auf S. 167 zu ausgewählten Aufgaben die Kommandos angegeben, die zur Lösung erforderlich sind. Zu vielen Aufgaben finden Sie dort in Abschnitt 6.4 eine Musterlösung - die sollten Sie allerdings nur in Notfällen oder zum Vergleich heranziehen!

Bitte arbeiten Sie die Aufgaben in ihrer Reihenfolge durch. Fortgeschrittene Anwender, die ihre Kenntnisse verfestigen oder noch etwas dazulernen wollen, können sich natürlich einzelne Aufgaben heraussuchen. Sie sollten jedoch auch die anderen Aufgaben des Abschnittes kennen, da die Aufgaben teilweise aufeinander aufbauen.

Voraussetzungen zum Durcharbeiten der Aufgaben:
- Sie müssen eine Kennung mit Shell-Erlaubnis besitzen;
- Sie müssen die Beispieldaten installiert haben (s. Abschnitt 6.1).

Aufruf der Shell und Prozeßkonzept

In diesem Abschnitt lernen Sie die ersten Sinix-Befehle kennen. Dabei ist das Prozeß-Konzept des Mehrprogramm-Betriebssystems Sinix von großer Bedeutung. Wir wollen es daher an den Anfang unserer Betrachtungen stellen.

Praktisch durch jede Aktion einer Kennung wird ein **Shellprozeß** erzeugt. Generell ist also die erste Aktion, die man zusammen mit dem Betriebssystem durchführt, der Shellprozeß zum Login.

Durch den Login wird nicht zwangsläufig eine **Kommando-Shell** (meist nur **Shell** genannt) erzeugt. Wenn Ihre Kennung nicht zu der Klasse "expert" gehört, die sich gleich nach dem Login in der Shell befindet, müssen Sie die Shell auf andere Art aufrufen. Im Standard-Menüsystem von Sinix V5.21 geben Sie ein Ausrufezeichen ein. In "COLLAGE" klicken Sie mit der Maus die Anwendung "sh" an. Daß Sie in der Shell sind, erkennen Sie am **Bereitzeichen** (Shell-Prompt), dem Dollarzeichen "$", das für Benutzer ohne Systemverwalterfunktionen als Eingabeaufforderung dient.

Geben Sie nun den Befehl

    who

ein. Schlagen Sie in der Kommandoauflistung für die Benutzerumgebung (Abschnitt 3.1) die Funktion dieses Befehls nach. (Wie man Kommandos eingibt, wurde in Abschnitt 2.1 erklärt.) Geben Sie nun die Befehle

    date

und

    tty

ein, und schlagen Sie ebenfalls deren Funktionen nach. - Ihr oben erzeugter Shell-Prozeß erzeugt für jedes der gerade eingegebenen Kommandos eine (**Sub-)Shell**, also eine **Ablaufumgebung**, die die Ausführung des jeweiligen Befehls steuert und danach sofort wieder beendet wird.

Kommandos können auch interaktiv ablaufen: Benutzer und System tauschen Daten durch Fragen und Antworten aus. Ein Beispiel ist der Befehl "passwd". Ändern Sie Ihr Kennwort mit dem Befehl

    passwd

Ihre erste Kommando-Shell beenden Sie durch Drücken der <ENDE>-Taste. Als Standardmenü-Benutzer erhalten Sie wieder das Standardmenüsystem, für Benutzer der Benutzerklasse "expert" wird die Sitzung beendet.

Rufen Sie nochmals die Shell auf! Sie erzeugen nun wieder einen neuen Shell-Prozeß, nämlich eine Kommando-Shell.

Kommando-Shells werden normalerweise mit dem Kommando "sh" erzeugt. Geben Sie also ein:

```
sh
```

Es scheint nichts passiert zu sein. Wenn Sie allerdings die <ENDE>-Taste betätigen, erscheint nochmals der Shell-Prompt - der Prompt der Shell, aus der Sie mit "sh" eine Kommando-Shell erzeugt haben.

Sie haben festgestellt, daß es auch Kommandos gibt, bei denen nichts Auffälliges geschieht. Sie können dem Betriebssystem aber "auf die Finger schauen", indem Sie das Kommando

```
ps
```

absetzen. Schlagen Sie das Kommando "ps" in der Kommando-Auflistung nach. Sie müssen nicht alles verstehen; es ist jedoch wichtig zu begreifen, daß Sinix intern genau über alle laufenden Prozesse Buch führt und jeden mit einer eindeutigen Nummer (Spalte "PID" für Process IDentification) versieht.

Eine Kommando-Shell taucht in der "ps"-Buchführung in der Spalte "COMMAND" als Befehl "sh" auf. Prozesse werden in dieser Spalte mit ihrem Kommando aufgelistet. Beispielsweise wird bei Menü-Benutzern in Sinix V5.21 zunächst eine Menü-Shell erzeugt (Kommando "msh"), die der Vaterprozeß der später erzeugten Subshells ist. Die Nummer des Terminals wird in der Spalte "TT" angezeigt. - Ihre Terminalnummer können Sie wieder mit dem Befehl

```
tty
```

herausfinden.
Ermitteln Sie nun mit Hilfe des "ps"-Kommandos die Prozeßnummer Ihrer aktuellen Shell und die Prozeßnummer Ihrer Login-Shell.

Kommen wir noch einmal zum Kommando "sh" zurück. Geben Sie wieder

```
sh
```

ein und danach

ps

Sie können jetzt die "Buchführung" des Betriebssystems weiterverfolgen: Eine neue Kommando-Shell wurde erzeugt und erscheint auch in der Auflistung des "ps"-Kommandos. - Erzeugen Sie mindestens fünf neue Kommando-Shells, und finden Sie jedesmal die neu für den Prozeß vergebene Prozeßnummer heraus. Drücken Sie nun die <ENDE>-Taste, und setzen Sie danach das "ps"-Kommando ab, um zu sehen, daß die letzte Kommando-Shell beendet wurde und daher aus der "Buchführung" verschwindet. Beenden Sie zum Schluß noch alle anderen Kommando-Shells.

Sie haben nun einen ersten Einblick in die Prozeßhierarchie von Sinix erhalten. Können Sie sich vorstellen, wie die Hierarchie für Ihre eigenen Prozesse aussieht, wenn Sie als Benutzer von "COLLAGE" (Kommando: "collage") die Shell aufrufen und einen Befehl (z. B. "date") absetzen? Versuchen Sie, sich diese Hierarchie graphisch zu veranschaulichen!

Kommandoeingabe

In den folgenden Abschnitten wird erwartet, daß Sie sich in der Shell befinden. Sie werden also nicht aufgefordert, eine Kommando-Shell aufzurufen.

1. Was passiert, wenn Sie ein Kommando falsch geschrieben haben (z. B. "daet" statt "date") und trotzdem "abschicken" (d. h. die Eingabe mit <RETURN> abschließen)?

2. Haben Sie schon festgestellt, ob Sinix Groß- und Kleinschreibung unterscheidet? Wenn nicht, geben Sie einmal "DATE" statt "date" ein. Testen Sie auch "Date".

3. Was passiert, wenn Sie nicht mit den Korrekturtasten, sondern mit den Pfeiltasten korrigieren und dann das Kommando mit <RETURN> "abschicken"?

Schalter und Objekte

4. Wie Sie im ersten Abschnitt gesehen haben, zeigt "who" alle am lokalen Rechner angemeldeten Benutzer an. Der Befehl kann Ihnen aber auch die Daten ausschließlich Ihrer Kennung anzeigen: Geben Sie

```
who am i
```

ein. Wann haben Sie den Login durchgeführt? Welche Nummer hat Ihr Terminal?

5. Sie erkennen am "who"-Kommando der vorigen Aufgabe, daß durch Zusätze zu dem bloßen Namen des Kommandos dessen Funktion variiert werden kann. Dies wird auch deutlich, wenn Sie den "ls"-Befehl in der Kommando-Auflistung genauer betrachten: Man kann mit Hilfe von sogenannten **Schaltern** die grundsätzlich gleich bleibende Wirkungsweise von Kommandos verändern.
Mit der Eingabe von "ls" listen Sie den Inhalt des aktuellen Dateiverzeichnisses. Geben Sie nun

```
ls -l
```

ein. Was ist der Unterschied zu "ls"? Testen Sie auch die Schalter "-x" und "-C"!

6. Schalter werden normalerweise mit einem "-" eingeleitet. "ps" bildet eine Ausnahme, weil Sie die Schalter des Kommandos sowohl mit als auch ohne "-" einleiten können: "ps a" und "ps -a" sind identisch (probieren Sie!).
Die in der Kommandoauflistung für "ps" angegebenen Schalter sind nur eine Auswahl aller möglichen. Hier wird der Sinn von Schaltern deutlich: Wenn man für jede einzelne Funktion ein neues Kommando einrichten würde, wäre das Betriebssystem völlig unüberschaubar.
Schalter können auch kombiniert werden, soweit dies einen Sinn ergibt. Testen Sie die folgenden Kombinationen, und identifizieren Sie jedesmal Ihre aktuelle Shell:

```
ps au
ps a
ps aux
ps ux
ps x
```

Vergleichen Sie die Ausgabe von

ps ux

und

ps u

Sie erkennen, daß die Informationen keine besonderen Unterschiede aufweisen. Der Schalter "x" ist eigentlich nur sinnvoll, wenn er mit "a" kombiniert wird.
Beim "ps"-Befehl ist die **Reihenfolge der Schalter** unwichtig: Testen Sie

ps au

und

ps ua

Es gibt jedoch andere Kommandos, bei denen das Betriebssystem bei falscher Reihenfolge der Schalter einen Syntaxfehler melden würde.

7. Ist das Kommando "ls -lx" sinnvoll? Begründen Sie Ihre Antwort!

8. Sie haben schon drei Kommandos mit Schaltern kennengelernt ("who", "ls" und "ps"). Der "ls"-Befehl kann jedoch auch mit Werten ergänzt werden, die keine Schalter sind: mit sog. **Objekten**. Beim "ls"-Kommando können Sie nämlich auch die aufzulistenden Dateien und Verzeichnisse genauer spezifizieren, indem Sie deren Namen hinter die eventuell angegebenen Schalter schreiben. Da diese Angaben nicht unbedingt gemacht werden müssen, sind sie in der allgemeinen Schreibweise des Kommandos, der Syntax, in Klammern gesetzt (Sie geben die Klammern natürlich nicht ein!). Eine Angabe, die nicht in runden Klammern steht, ist obligatorisch.
Lassen Sie sich den Inhalt der Dateiverzeichnisse "/usr" und "/etc" auflisten, während Sie sich noch in Ihrem Stamm-Dateiverzeichnis befinden.

Prozeßstatus

9. Mit dem "ps"-Befehl (s. Aufgabe 6) können Sie auch kontrollieren, was andere Benutzer machen. Geben Sie immer "ps aux" ein, um eine möglichst umfassende Information zu erhalten. Sie können

den Schalter "x" auch weglassen, wenn Sie an den Prozessen des Betriebssystems selbst nicht interessiert sind. Zur Übung suchen Sie bitte die folgenden Prozesse mit Hilfe des Befehls "ps aux" heraus:

a) Programme zur Druckersteuerung: "/usr/spool/spooler/lp..."

b) die Vaterprozesse aller anderen Prozesse: PID 0, 1 und 2

c) Programme zur Terminalinitialisierung: "... (getty)"

Universen

Universen sind herstellerspezifische Ablaufumgebungen für Kommandos und Programme innerhalb des Betriebssystems Sinix V5.xx (vgl. hierzu Abschnitt 2.2). Da die Unterschiede im Kommandoaufruf zwischen den Universen praktisch kaum relevant sind, soll dieser Themenkreis nur sehr kurz angesprochen werden. Sie sollten nur darauf achten, ob das jeweilige Kommando auch in Ihrem Universum vorhanden ist. Weil die folgenden Aufgaben im AT&T-Universum getestet wurden, empfehlen wir Ihnen, **ab sofort in diesem Universum zu arbeiten.**

10. Nur bis einschließlich Sinix V5.23: Testen Sie die Funktion der Kommandos "att" und "sie". Vergleichen Sie dann die Ausgabe der Kommandos "att ls" und "sie ls" (möglichst in einem "vollen" Dateiverzeichnis, z. B. "/bin")!

11. Lassen Sie sich das aktuelle Universum ausgeben.

Benutzerumgebung

Einige Kommandos für die Benutzerumgebung haben Sie bereits im ersten Abschnitt (Aufruf der Shell und Prozeßkonzept) kennengelernt: "who", "date", "tty", "passwd" und "ps". Drei weitere werden Sie zum Lösen der folgenden Aufgaben verwenden.

12. Jeder Kennung ist nicht nur ein eindeutiger Kennungsname zugeordnet, sondern auch eine damit verbundene unverwechselbare Benutzernummer sowie die Gruppenidentifikation in Form des Gruppennamens und der Gruppennummer. Welches Kommando informiert Sie über diese Werte? Testen Sie den Befehl!

13. Durch das Kommando "ls -l" haben Sie einen ersten Einblick in das Sicherheitskonzept erhalten: Es gibt in Sinix gestufte Zugriffsberechtigungen. Eine weitere Möglichkeit, unbefugtes Eindringen in das System zu verhindern, liegt im Paßwortschutz der Kennungen. Versuchen Sie, das Kennwort einer anderen Kennung (z. B. "admin") zu ändern!

14. Wechseln Sie in der Shell Ihre Benutzerkennung (z. B. zu "gast", deren Kennwort Sie ggf. beim Systemadministrator erfragen können).

15. Sie wollen nun wissen, wieviel Speicherplatz das Beispiel-Dateiverzeichnis "Bsp" belegt, da Sie nicht über unbegrenzten Platz verfügen und nicht "über Ihre Verhältnisse leben" wollen. Suchen Sie den entsprechenden Befehl aus der Kommandoauflistung für die Benutzer-Umgebung heraus, und geben Sie ihn ein.

Sonderfunktionen der Shell

Lesen Sie zu den folgenden Aufgaben bitte Abschnitt 3.7 über die "Sonderfunktionen der Shell (II)".

16. Erzeugen Sie mit dem "echo"-Befehl folgende Ausgaben:

a) "$

b) '`"*

c) ???

d) *

e) ¦¦¦\\\

17. Wenn der Inhalt von Dateiverzeichnissen zu groß für den Bildschirm ist (wenn die Namen "durchrauschen"), verlieren Sie die ersten Einträge. Auch der Befehl "ls -x" hilft hier nicht

immer - testen Sie ihn z. B. für "/bin" und "/etc"! Wenn Sie den Inhalt eines Dateiverzeichnisses mit den Zugriffsberechtigungen auflisten, bringt der Schalter "-x" überhaupt nichts - testen Sie auch das. Dies ist ein Anlaß, ein weiteres Kennzeichen von Sinix/Unix vorzustellen, nämlich die sogenannte "Pipe".
Mit dem Kommando "pg" (im AT&T-Universum; "more" im Siemens-Universum) kann man sich Dateien bildschirmweise anschauen. Sie übergeben die Ausgabe des "ls"-Kommandos direkt an "pg", indem Sie ein "Pipe"-Zeichen dazwischen schreiben:

```
ls ¦ pg
```

Testen Sie diese "Pipe" mit den Schaltern "-x" und "-l" für den "ls"-Befehl anhand der Verzeichnisse "/etc" und "/bin"!

18. Können Sie mit nur einer Kommandozeile zählen lassen, wieviele Verzeichniseinträge das Dateiverzeichnis "/bin" enthält?
Tip: Kombinieren Sie "ls" mit "wc".

19. Kombinieren Sie "who" und "wc", um als Ausgabe die Anzahl der aktiven Benutzer zu erhalten.

20. Sind die folgenden Kommandos sinnvoll? Begründen Sie Ihre Antwort!

```
pg prog ¦ ced
ls ¦ cp Ordner
sh ¦ pg
```

21. Eine weitere Möglichkeit, Arbeit zu sparen, besteht in der Verwendung von sogenannten "Matchcodes", "Wildcards" bzw. "Jokern", die Teile von Eintragsnamen abkürzen (vgl. Abschnitt 2.4.1). Wechseln Sie in das Dateiverzeichnis "/bin", und listen Sie dort alle Dateinamen, die

a) mindestens ein "e" enthalten,
b) mit "c" beginnen,
c) mit "y" aufhören,
d) die Zeichen "e" und "m" enthalten,
e) nur aus vier **Buchstaben** bestehen (also nicht "abc4").

Tip: Verwenden Sie zum Testen mit dem "ls"-Befehl den Schalter "-d".

22. Mit welchen Matchcodes können Sie **alle** Namen in einem Dateiverzeichnis erfassen, nicht jedoch "." und ".."? ("*" erfaßt nicht die mit einem "." beginnenden Einträge!)

Tip: Verwenden Sie zum Testen mit dem "ls"-Befehl den Schalter "-d".

Das Bewegen im Dateisystem

Für diesen Übungsabschnitt wiederholen Sie bitte den Abschnitt 2.3 über "Das Sinix-Dateisystem".
Bildlich können Sie sich unter dem Dateisystem ein System aus Aktenschränken, Aktenordnern und Dokumenten vorstellen: Ein Schrank (Dateiverzeichnis oder Directory, Abkürzung: DVZ) enthält Ordner (DVZ der nächsttieferen Stufe), die wiederum Dokumente enthalten (Dateien), in denen die eigentlichen Daten (z. B. ein Geschäftsbrief) zu finden sind. An diese dreistufige Hierarchie muß man sich in Sinix jedoch nicht halten, da man Dateiverzeichnisse theoretisch in beliebig vielen Stufen anlegen kann. Jeder **Verzeichniseintrag** (Datei oder Dateiverzeichnis) wird mit einem bestimmten Namen ("Pfadnamen") angesprochen. Wenn Sie nicht mehr ganz sicher sind, was absolute und relative Pfadnamen sind, schlagen Sie bitte in Abschnitt 2.3.2 nach.

23. Wie lautet der absolute Pfadname Ihres Stamm-Dateiverzeichnisses?

Tip: Schlagen Sie das Kommando unter "Benutzer-Umgebung" nach.

24. Dateiverzeichnisse können Dateien oder wiederum Dateiverzeichnisse als "Einträge" enthalten. Sie können natürlich auch leer sein. Listen Sie den Inhalt Ihres Stamm-Dateiverzeichnisses auf.

Tip: Schlagen Sie unter "Bearbeitung des Dateisystems" nach.

25. Auch ein leeres Dateiverzeichnis ist nicht völlig leer. Das zeigt Ihnen das "ls"-Kommando mit dem Schalter "-a". Wechseln Sie in das Dateiverzeichnis "Leer" der Beispieldaten. Was ist der Unterschied zwischen den Kommandos "ls" und "ls -a"?

26. a) Das Kommando "cd" dient zum Wechseln des aktuellen Dateiverzeichnisses (der "working directory"). Im oben angesprochenen bildlichen Vergleich würden Sie zu einem anderen Aktenschrank gehen oder einen anderen Ordner aufschlagen. Wechseln Sie zum Dateiverzeichnis "/usr", und listen Sie dessen Inhalt auf.

b) Wie können Sie nun zu Ihrem Stamm-Dateiverzeichnis zurückwechseln, ohne daß Sie dessen Pfadnamen angeben müssen?

27. Wechseln Sie zum Ursprungs-DVZ des gesamten Dateisystems, und listen Sie dessen Inhalt auf. Können Sie "noch höher hinaus"?

28. Durch die Verwendung von relativen Pfadnamen kann man u. a. Schreibarbeit sparen. Was ist von den folgenden Einträgen absoluter, was relativer Pfadname?

```
prog
/usr/gast
usr/ordner1/doku
/etc/passwd
etc/fsck
/
usr/ordner2/dateivz/nix
../dvz/dvz/dvz/nochmal
bin
```

29. Angenommen, Sie befinden sich im Dateiverzeichnis mit dem Namen "/usr1/maier".

a) Wie sieht dann der relative Pfadname zu den folgenden absoluten Pfadnamen aus?

```
/usr1/maier/texte
/usr1/maier/ordner1/kurse
/usr1/maier/Bsp/programme
/usr1/maier2/susi
/usr1/maier2/Dokumente/briefe
/usr1
/usr
/usr1/maier
```

b) Wie lautet dann der absolute Pfadname zu den folgenden relativen Pfadnamen?

```
briefe/susi
./ottokar/rechnung
.
../DB
../../usr/bin/hit
DB/datenbank
```

30. Prüfen Sie, ob die folgenden Dateiverzeichnisse existieren, indem Sie dorthin wechseln, den Inhalt auflisten und zwischendurch immer wieder zurück zu Ihrem Stamm-Dateiverzeichnis gehen:

```
/
/usr/menus/app/control
/usr/att/usr/lib/hit/printer
/usr/lib
/usr/lib/hit/printer
/usr/att
/usr/sie_root
/.lib
/etc
/bin
/usr/bin
```

31. Nur der Systemverwalter (die Kennungen "root" und "admin") hat universellen Zugriff auf sämtliche Einträge des Dateisystems. Als normaler Anwender müssen Sie sich an die Zugriffsrechte halten, die innerhalb des Systems gültig sind. Schlagen Sie unter dem "ls"-Befehl nach, welcher Schalter die Einträge mit ihren Zugriffsrechten auflistet. Geben Sie den Befehl im Dateiverzeichnis "Bsp" ein. Interpretieren Sie die Zugriffsrechte (Erklärungen finden Sie unter dem Kommando "chmod"). Listen Sie den Inhalt der Dateiverzeichnisse der vorigen Aufgabe mit allen Zugriffsrechten auf.

32. Versuchen Sie - wenn Sie nicht unter der Kennung "root" oder "admin" arbeiten -, im DVZ "/usr/bin" die Datei "sleep" zu löschen! Prüfen Sie dann deren Zugriffseinstellungen.

Zugriffsrechte und Eigentümer

33. Zu welcher Gruppe gehört das Dateiverzeichnis "Leer"?

34. a) Nehmen Sie dem Eigentümer alle Zugriffsrechte auf "Leer", während Sie sich in "Leer" befinden.

b) Versuchen Sie, aus "Leer" eine Stufe nach "oben" zu wechseln. Welches Problem entsteht nun und warum? (Es hat nichts damit zu tun, daß "Leer" leer ist!)

c) Geben Sie dem Eigentümer wieder Lese- und Schreiberlaubnis für das Dateiverzeichnis "Leer".

d) Wechseln Sie aus "Leer" eine Stufe "nach oben".

35. Ändern Sie im gesamten Unterverzeichnis "Bsp" die Zugriffsrechte: Lesen, Schreiben, Ausführen für den Eigentümer; Lesen, Ausführen für die Gruppe; Ausführen für die anderen Benutzer. Kontrollieren Sie das Ergebnis, indem Sie sämtliche Zugriffe auflisten und bildschirmweise ansehen.

Sichten von Dateien

36. a) Dateien können mit dem Kommando "cat" am Bildschirm ausgegeben werden. Lassen Sie sich die Datei "kurz" am Bildschirm ausgeben.

b) Wenn man jedoch eine längere Datei damit anzeigen will, "rauscht" deren Inhalt ohne anzuhalten durch. Testen Sie dies anhand der Datei "/etc/rc"!

37. Eine bildschirmweise Anzeige bietet das Kommando "pg" ("more" im Siemens-Universum). Testen Sie es anhand der Datei "/etc/passwd"!

38. a) Wenn Sie sich in einer Datei frei bewegen wollen - z. B. zeilen- oder bildschirmweise wieder nach "oben" -, benötigen Sie einen Editor, z. B. den CED. Laden Sie die Datei "/etc/passwd" in den Editor. (Den CED können Sie mit der <ENDE>-Taste wieder verlassen.)

b) Wechseln Sie in das Dateiverzeichnis "Bsp/Texte", und sehen Sie sich die darin abgelegten Dokumente an. Versuchen Sie, die Datei "kopf" in den CED zu laden! ("kopf" ist eine HIT-Formatiertabelle und kann vom CED nicht bearbeitet werden.)

39. Legen Sie eine Datei "nixda!" an, und entziehen Sie sich (dem Eigentümer) daraufhin das Leserecht. Versuchen Sie nun, diese Datei wieder in den CED zu laden.

Umlenkung des Standard-Ausgabe-Kanals

40. Legen Sie von den Einträgen des Dateiverzeichnis "Bsp/Inf" ein Inhaltsverzeichnis an. Laden Sie das so erzeugte Dokument in den CED, und versehen Sie es mit einer Überschrift (z. B. "Alle Datenbank-Dateien"). Was ist der normale Standard-Ausgabe-Kanal?

Tip: Es wäre sehr zeitraubend, die Datei manuell zu füllen; Sinix nimmt Ihnen einen Teil der Arbeit ab, wenn Sie die Ausgabe des "ls"-Kommandos direkt in eine Datei umleiten (vgl. Abschnitt 2.4.3).

41. a) Erstellen Sie mit Hilfe des CED die folgenden Dokumente: "adresse" (mit Ihrer eigenen Adresse in Briefkopfform), "abschied" (mit einer formellen Verabschiedung für einen Brief, z. B. "mit vielen Grüßen"), "ciao" (mit einem freundlichen "Ciao" als Brieffuß), "einladung" (Einladungstext zu einem Fest mit Ihren Freunden) und "frech" mit dem Inhalt, daß Sie Ihren Chef um eine Gehaltserhöhung um 20 % bitten.

b) Sie kennen das Kommando "cat" schon als (unkomfortablen) Befehl zum Ansehen von Dateien. Eine nützlichere Funktion wird durch die Kombination von "cat" mit ">" bzw. ">>" erreicht (Umleitung des Standard-Ausgabe-Kanals). In Aufgabe a) haben Sie verschiedene Dateien ("Textbausteine") erstellt, die nun zu sinnvollen Dokumenten verbunden werden sollen:

adresse + frech + abschied = gehalt
adresse + einladung + ciao = fest

Erzeugen Sie zuletzt noch ein Dokument mit einem Postskriptum (Inhalt: Zeiten, zu denen Sie telefonisch zu erreichen sind), das Sie an beide Dateien ("gehalt" und "fest") anhängen. Prüfen Sie, ob die Befehle die richtigen Ergebnisse erzeugt haben.

42. Ein Dateiverzeichnis ist eigentlich eine besondere Datei. Dies wird deutlich, wenn man den Befehl "cat DVZ > Neudatei" eingibt. Wählen Sie ein geeignetes Dateiverzeichnis, und schauen Sie sich "Neudatei" an!

Anlegen, Versetzen und Löschen von Dateien und DVZ

Sie können sich im Dateisystem schon sicher bewegen. In diesem Abschnitt greifen Sie - wie bereits in den letzten Aufgaben - verändernd in das Dateisystem ein, indem Sie Dateien und Verzeichnisse anlegen, versetzen und löschen.

43. Um von vornherein zu verhindern, daß Ihre Dateien und Verzeichnisse ungeordnet und unübersichtlich vorliegen, richten Sie sich zunächst eine geeignete Ablagestruktur ein. Legen Sie unter Ihrem Stamm-DVZ also die folgenden Ordner (Dateiverzeichnisse) an:

- "Ablage": zu erledigende Aufgaben (z. B. zu überarbeitende Datei)
- "Briefe": Briefe und Standardmitteilungen für die elektronische Post ("Mail")
- "Informix": Daten für das Datenbanksystem INFORMIX
- "Loesungen": für Lösungen der Übungsaufgaben
- "Muell": wahrscheinlich nicht mehr benötigte Dateien
- "Sicher": zur Ablage von Sicherheitskopien
- "Siplan": Dateien des Tabellenkalkulationssystems SIPLAN
- "Software": z. B. für nützliche oder lehrreiche Shell-Prozeduren
- "Testen": zur Ablage von Testdaten
- "Texte": zur Ablage von Texten
- "Tips": Tips für die Shell

Anmerkung: Standard-Archive für das Textsystem HIT werden normalerweise beim ersten Aufruf von HIT eingerichtet.

44. a) Löschen Sie das noch leere Dateiverzeichnis "Testen", und legen Sie ein neues mit dem Namen "Tester" an.

b) Wie hätten Sie die Aufgabe a) komfortabler durchführen können?

Tip: Schauen Sie sich das "mv"-Kommando an.

45. Angenommen, Sie haben (z. B. mit "du -s") herausgefunden, daß Sie zuviel Speicherplatz verbrauchen. In diesem Fall sollten Sie versuchen, im Beispiel-Dateiverzeichnis "Bsp" "aufzuräumen", also alle unnützen Dateien und Verzeichnisse zu löschen. Löschen Sie die folgenden Einträge, nachdem Sie sich die (meist sinnlosen) Inhalte angesehen haben:

a.out (beim Anschauen können Schmierzeichen erscheinen)
wegmit
kurz

```
Krempel1
Krempel2
Krempel3/Mist
```

Tip: Wenn Sie komfortabel (aber unter Umständen riskant) ein Dateiverzeichnis mit seinen Unterdateiverzeichnissen löschen wollen, halten Sie in der Kommandoauflistung nach den Schaltern "-r" und "i" des betreffenden Löschbefehls Ausschau.

46. Sind die folgenden Kommandos sinnvoll? Begründen Sie Ihre Antwort!

```
mv prog prog
mv * datei
mv *.bak *.sik
```

47. a) Wechseln Sie in das Dateiverzeichnis "Bsp/Texte", und benennen Sie die Datei "kopf" in "hitkopf" um. Benennen Sie die Datei "mottos" in "leitsaetze" um.

b) Versetzen Sie die Dateien "hitkopf", "mathe" und "lernen" in das Verzeichnis "Texte" unterhalb Ihres Stamm-DVZ.

c) Die Dateien "p1", "p2" und "p3" sollen in das Dateiverzeichnis "Software" (ebenfalls unterhalb Ihres Stamm-Dateiverzeichnis) verlegt werden.

48. Löschen Sie das Dateiverzeichnis "Texte" unter "Bsp".

49. Versetzen Sie den gesamten Inhalt von "Bsp/DVZ1" in das Verzeichnis "Muell" Ihres Stamm-Dateiverzeichnisses.

50. a) Legen Sie mit Hilfe des Texteditors CED unter Ihrem Stamm-Dateiverzeichnis die Datei "prog" mit folgendem Inhalt an:

```
ls
cd /etc
ls -x ¦ pg
echo ENDE
```

b) Geben Sie das folgende Kommando ein:

```
sh prog
```

Sie haben damit Ihre erste - noch recht einfache - Shell-Prozedur gestartet. Beobachten Sie, was passiert.

Tip: Woher bezieht die Shell nun ihre Eingaben?

51. Versetzen Sie die Datei "prog" in das Dateiverzeichnis "Tester".

Suchen von Dateien und Dateiverzeichnissen

52. Lassen Sie sich **alle** Dokumente unterhalb Ihres Stamm-Dateiverzeichnisses mit Hilfe des "find"-Kommandos anzeigen.

53. Suchen Sie unterhalb Ihres Stamm-Dateiverzeichnisses
a) alle Dateien mit dem Namen "brief".
b) alle Dateien, die mindestens ein "e" enthalten.

54. Testen Sie das folgende Kommando, und schlagen Sie den Befehl "file" im Handbuch nach:

```
file `find . -print` ¦ pg
```

(Sicherheits-)Kopien

55. Legen Sie von der Datei "prog" eine Kopie "prog.sik" an.

56. Erstellen Sie eine Kopie des gesamten Verzeichnisses "Bsp/Informix" mit dem Namen "FIRMA". Andere Benutzer können auf diese Daten mit Hilfe von INFORMIX nur zugreifen, wenn sie Lese- und Schreibrechte besitzen. Vergeben Sie also die entsprechenden Rechte.

Tip: Im Siemens-Universum erstellt man mit dem "copy -r"-Befehl rekursive Kopien, im AT&T-Universum muß man "find" mit "cpio -pd" kombinieren.

57. a) Kopieren Sie sich die Datei "/bin/find" in Ihr Stamm-DVZ, und komprimieren Sie sie. Wieviele Bytes wurden gespart?
b) Löschen Sie die komprimierte "find"-Datei.

58. a) Erstellen Sie eine Sicherheitskopie "firma" des gesamten Dateiverzeichnisses "Inf" (es liegt unterhalb von "Bsp") mit Hilfe des "cpio"-Kommandos, und legen Sie sie im Dateiverzeichnis "Sicher" ab.
b) Komprimieren Sie die soeben erstellte Sicherheitskopie.
c) Spielen Sie die Sicherheitskopie aus b) im DVZ "Tester" als "Informix" ein (Entkomprimieren nicht vergessen).
d) Löschen Sie das Dateiverzeichnis "Tester/Informix".

59. Sind die folgenden Kommandos sinnvoll? Begründen Sie Ihre Antwort!

```
cp prog prog
cp alle* *
cp a_liste.* b_liste.*
ls -R ¦ cpio -o > Ausdatei
copy -r DVZ Datei
```

Diskettenbearbeitung

60. Prüfen Sie, wieviel Speicherplatz Sie belegt haben. Informieren Sie sich beim Systemverwalter oder einem anderen Benutzer, wieviel Speicherkapazität die Disketten für Ihre Anlage haben, und formatieren Sie so viele Disketten, wie Sie voraussichtlich brauchen. Speichern Sie dann alle Ihre Daten ab. Kontrollieren Sie den Vollzug, indem Sie sich das ausführliche Inhaltsverzeichnis Ihrer Diskette(n) ansehen.

61. Warum ist der Befehl "far xvr Datei" nicht sinnvoll?

62. Legen Sie ein Dokument "inhalt" an, in dem der Inhalt der Diskette steht, und hängen Sie diese Datei an die Daten auf der letzten Diskette an. Löschen Sie dann die Datei "inhalt".

63. a) Erstellen Sie eine Sicherheitskopie des Dateiverzeichnisses "Bsp/Inf" auf Diskette.
    b) Löschen Sie das DVZ "Bsp/Inf", und spielen Sie es neu von der Diskette ein. (Nehmen Sie z. B. an, daß durch einen Systemabsturz Daten in der Datenbank verloren gegangen sind.)

Druckausgabe von Dokumenten

64. Informieren Sie sich über die Druckergruppen und die Auftragslage.

65. Lassen Sie die Dokumente im Dateiverzeichnis "Korrespondenz" mit nationalem Zeichensatz und der Zeichenbreite 12 Zeichen/Zoll ausdrucken.

66. Wieviele Zeilen und Wörter enthalten die gerade ausgedruckten Dokumente? Erstellen Sie eine Datei namens "info", in der die Lösung dieser Aufgabe und das Ergebnis stehen, und drucken Sie "info" aus.

67. a) Erteilen Sie einen Druckauftrag mit Namen "test", mit dem Sie die Datei "mathe" ausdrucken. Wenn Sie nicht auswendig wissen, wo sie steht, müssen Sie sie mit "find" suchen. Löschen Sie dann den gerade abgeschickten Druckauftrag wieder!

    Tip: Wenn Sie das Löschen von Druckaufträgen üben wollen, empfiehlt es sich, den Drucker vorher "Off-line" zu schalten, damit der Auftrag nicht beendet ist, bevor Sie ihn gelöscht haben.

    b) Lassen Sie das Dokument "mathe" mit dem Befehl "pr" druckaufbereiten (60 Zeilen pro Seite, eine passende Überschrift, 2 Spalten), und leiten Sie die Ausgabe direkt an einen Drucker Ihrer Wahl weiter.

Hintergrundprozesse und fremde Prozesse

68. Ein bisher noch nicht genutzter Vorteil des Prozeß-Konzeptes ist die Möglichkeit des Abschickens von sog. Hintergrundprozessen (vgl. Abschnitt 2.4.4). Der Prozeß läuft dann parallel zur aktuellen Shell ab, man kann also ungestört weiterarbeiten.

    Erzeugen Sie mit Hilfe des "find"-Befehls ein Dokument, in dem alle Einträge unter Ihrem Stamm-Dateiverzeichnis mit ihren Zugriffsberechtigungen aufgelistet sind. Legen Sie den Prozeß mit dem "find"-Befehl in den Hintergrund.

    Tip: Folgende Schalter sind relevant: "-exec" für den "find"-Befehl und "-ld" für den "ls"-Befehl. Denken Sie daran, daß Sie die Ausgabe umleiten müssen!

69. a) Das Kommando "sleep Zahl" legt die aktuelle Shell für "Zahl" Sekunden still. Was passiert, wenn Sie den Befehl "sleep 10" in den Hintergrund legen?

    b) Starten Sie einen Hintergrundprozeß mit dem Befehl "sleep 1000 &". Zerstören Sie diese Subshell.

70. Schreiben Sie nur diejenigen Zeilen der Ausgabe des "ps aux"-Kommandos in eine Datei, in denen die Zeichenkette "root" vorkommt.

71. Versuchen Sie, einen Prozeß eines anderen Benutzers zu löschen!

Kommunikation in Sinix-Systemen

Die hier kurz angesprochenen Kommandos, die der Kommunikation sowohl innerhalb des Betriebssystems als auch innerhalb eines LAN (Local Area Network) dienen, zählen teilweise zur Anwendungssoftware (CCP-LAN1/REMOS ab V3.xx) und sind nicht unbedingt auf Ihrem Rechner installiert. Außerdem müssen Sie zum Testen der LAN-Kommandos "rlogin" und "ftp" eine Kennung an einem vernetzten Partnerrechner besitzen. Um die Kommunikationskommandos testen zu können, empfiehlt es sich, mit anderen Benutzern zu kooperieren oder sich eine weitere Kennung zuteilen zu lassen.

72. Ein sehr nützliches Sinix-Kommando ist das "talk"-Kommando, das auch im LAN eingesetzt werden kann. Geben Sie ein:

    talk Benutzerkennung

    Der angesprochene Benutzer am eigenen Rechner erhält eine Nachricht und kann mit demselben Befehl die Verbindung herstellen. Wenn Sie eine Kennung an einem anderen Rechner erreichen wollen, geben Sie folgendes ein:

    talk Benutzerkennung@Rechnername

    Testen Sie beide Varianten!

73. Auch die normale elektronische Post wird durch Shell-Kommandos gesteuert. Lassen Sie den gesamten Inhalt Ihres Stamm-Dateiverzeichnisses auflisten, und senden Sie diese Aufstellung direkt (in einer Kommandozeile!) an eine Kennung Ihrer Wahl

    a) auf Ihrem Rechner.
    b) auf einem anderen Rechner.
    c) Schicken Sie die Dokumente unter dem Dateiverzeichnis "Korrespondenz" an die Kennungen "gast", "mgast", "admin" und an eine weitere Ihrer Wahl.

    Tip: Schlagen Sie unter dem Kommando "mail" nach.

74. Schicken Sie einer Kennung Ihrer Wahl einen Gruß auf den Bildschirm (nicht mit "talk" und nicht mit "mail").

Tip: Sie können Ihren eigenen Bildschirm für Nachrichten anderer Benutzer mit dem Befehl "mesg n" sperren.

75. Geben Sie die folgenden Kommandos ein, und beobachten Sie, was passiert:

```
mesg
ls -l `tty`
mesg n
ls -l `tty`
mesg y
ls -l `tty`
```

76. Testen Sie die folgenden LAN-Befehle:

rwho, ruptime, rlogin (ferner Rechnername), ftp (ferner Rechnername)

Tip: Sie beenden "rlogin" und "ftp" mit der <ENDE>-Taste.

Kommandos als Programmdateien

77. Sie haben nun schon häufig Kommandos eingegeben. Wir wollen den grundsätzlichen Mechanismus dabei nochmals verdeutlichen: Ein Kommando ist nichts anderes als eine ausführbare Programmdatei in der Programmiersprache C. Wenn Sie den Namen eines Kommandos eingeben, sucht Sinix danach an bestimmten Stellen im Dateisystem.

a) Eines dieser automatisch abgesuchten Dateiverzeichnisse ist das Dateiverzeichnis "/bin". Suchen Sie dort nach der Datei "ls". Kopieren Sie sich diese Datei in Ihr Stamm-DVZ. Wie groß ist diese Datei? Welche Zugriffsrechte sind dafür gültig? Benennen Sie die Datei in "lister" um. Geben Sie "lister" ein! Was passiert? Löschen Sie die Datei wieder.

b) Um Ihnen die Herkunft der Kommandos einmal zu demonstrieren, wollen wir nun ein eigenes C-Programm betrachten. Wechseln Sie in das Dateiverzeichnis "Bsp/Pgm". Schauen Sie sich die Datei "c-compiler" an. Sie enthält den Befehl zum Kompilieren des Programms "hallo.c", das Sie sich ebenfalls ansehen

sollten. Kompilieren heißt, daß das in Klartext vorliegende Programm in für die Maschine verständlichen Code umgewandelt wird, so daß es direkt ausführbar ist.
Starten Sie die Übersetzung mit dem Befehl "sh c-compiler". Prüfen Sie, ob eine ausführbare Datei namens "hallo" erstellt wurde. Geben Sie "hallo" ein. Was ist passiert?

Automatisierung mit Hilfe von Shell-Prozeduren

Lesen Sie zu den folgenden Aufgaben bitte Kapitel 4 über "Die Shell als Programmiersprache".

78. Erstellen Sie sich eine Datei ".profile" direkt unter Ihrem Stamm-Dateiverzeichnis mit den folgenden Funktionen:

- das Bereitzeichen soll ein invers dargestelltes "?" sein;
- Datum und Uhrzeit sollen ausgegeben werden;
- alle gerade arbeitenden Benutzer werden angezeigt;
- Ihre Speicherplatzbelegung wird angezeigt.

Führen Sie dann einen neuen Login durch.

Tip: Die Variable, die das Bereitzeichen enthält, heißt "PS1" (schlagen Sie im Abschnitt 2.4.5 nach!). Wenn Sie vor das Fragezeichen die Sequenz "\033[07m" und dahinter "\033[0m" in Anführungsstrichen schreiben, wird es invers angezeigt.

79. Das erste Kommando in Ihrer Prozedur ".profile" aus Aufgabe 78 soll nun das Kommando "sh" sein. Beenden Sie dann Ihre Sitzung, und führen Sie einen neuen Login durch. Was ist passiert?

80. Schreiben Sie eine Prozedur "antwort", die so lange eine Eingabe anfordert, bis entweder "j" bzw. "J" oder "n" bzw. "N" eingegeben wurde. Im Falle von "j"/"J" soll der Ende-Status "0" geliefert werden, bei "n"/"N" der Ende-Status "1".
Diese Prozedur können Sie dann wie folgt in andere Prozeduren einbauen:

```
if antwort
   then ...
   else ...
fi
```

Tip: Der Befehl zum Abbruch der Prozedur lautet "exit".

81. a) Schreiben Sie eine Prozedur "erste.scr", die Sie begrüßt und Ihnen die momentan aktiven Benutzer anzeigt. Danach soll die Prozedur anhalten und - nachdem Sie auf <RETURN> gedrückt haben - Ihnen die Systemauslastung und das aktuelle Datum anzeigen.

    Tip: Verwenden Sie die Kommandos "echo" und "read".

    b) Binden Sie in die Prozedur aus Aufgabe a) ein Frage-Modul ("while ...") ein, so daß nach jeder Information die Frage kommt: Wollen Sie weitermachen? (j=ja/n=nein)

    Danach muß eine "if"-Abfrage aufgerufen werden, die prüft, ob "j" eingegeben worden ist.

    Tip: Schauen Sie sich Aufgabe 80 an!

    c) Verändern Sie die gerade geschriebene Prozedur so, daß am Ende gefragt wird, ob die Belegung der Plattenbereiche (Systemauslastung) und die eigene Speicherbelegung ausgedruckt werden sollen (versehen Sie die Ausgabe mit korrekten Überschriften). Realisieren Sie den Druckmodul, indem Sie

    ca) die Ausgabe der Befehle in eine Datei schreiben, die auf Benutzerabfrage ausgedruckt werden kann (löschen nicht vergessen).

    cb) die Ausgabe direkt an den Druckbefehl weiterleiten.

    d) Die Druckausgabe in der vorhergehenden Aufgabe soll in eine Datei geschrieben, dann am Bildschirm ausgegeben werden und auf Wunsch seitenweise formatiert am Drucker ausgedruckt werden können.

    Tip: Sehen Sie sich das "pr"-Kommando an.

82. Schreiben Sie eine Prozedur "ist_aktiv", die meldet, ob ein Benutzer am Rechner arbeitet. Das Format des Aufrufs soll "ist_aktiv <Kennung>" sein.

83. Schreiben Sie eine Sicherungsprozedur mit folgenden Eigenschaften:

- Der Text

    "Sicherung vom <Datum> der Kennung <Kennung>;
    Pfad des aktuellen Dateiverzeichnisses: <Pfad>"

    wird in eine Datei "SICHER.inf" geschrieben, wobei die Werte für <Datum>, <Kennung> und <Pfad> automatisch eingesetzt werden.

- Alle Dateien unterhalb von <Pfad> werden in eine Datei "SICHER.dat" geschrieben, die gepackt wird.
- Die Dateien "SICHER.inf" und "SICHER.dat.z" werden auf Diskette gesichert.

84. Für INFORMIX-Kundige: Modifizieren Sie die Beispiel-Prozedur aus Abschnitt 4.6 so, daß aus der Tabelle "auftrag" die Aufträge der abzufragenden Auftragsnummer mit ihrem Brutto-Auftragswert am Bildschirm angezeigt werden und auf Wunsch ausgedruckt werden können.

6.3 Lösungshinweise

In der folgenden Tabelle sind für ausgewählte Aufgaben die zur Lösung benötigten Kommandos angegeben. Die Schalter und die weitere Syntax finden Sie in der Kommando-Auflistung.

Aufgabe	Kommandos, die zur Lösung benötigt werden
11.	universe
32.	rm, ls
33.	ls
34.	cd, chmod, ls
43.	mkdir
45.	ls, pg, rm
57.	cp, pack, ls, rm
60.	du -s, /etc/flformat (V5.4: /etc/format), tar
64.	lpr
66.	wc, ced, lpr
67.	lpr, find, pr, ¦
74.	write

6.4 Lösungen zu ausgewählten Übungsaufgaben

Die folgenden ausführlicheren Musterlösungen zeigen im allgemeinen nur eine von mehreren möglichen Lösungsvarianten auf.

Schalter und Objekte

7. Der Schalter "-l" listet sämtliche Einträge im Dateiverzeichnis **zeilenweise** (also ein Eintrag pro Zeile) mit ausführlicher Information auf. Hier werden z. B. der Name des Eigentümers der Datei, Zugriffsrechte, Erstellungs- und Änderungsdatum angegeben. Der Schalter "-x" dagegen ist sinnvoll, wenn nur die Namen der Einträge (ohne Zusatzinformation) im übersichtlichen Spaltenformat ausgegeben werden sollen. Das Kommando "ls -lx" ist also nicht sinnvoll.

Sonderfunktionen der Shell

18. ls -a ¦ wc -l

20. pg prog ¦ ced:
Der CED liest seine Eingaben von der Tastatur, nicht von anderen Programmen.

ls ¦ cp Ordner:
Das Kommando "cp" verarbeitet keine Eingaben, sondern kopiert Dateien, die beim Aufruf des Befehls angegeben werden müssen. Außerdem ist die Syntax des "cp"-Befehls hier etwas durcheinander geraten.

sh ¦ pg:
Das Kommando "sh" erzeugt keine Ausgabe - also kein Verarbeitungsergebnis -, sondern ist ein interaktives Programm. "pg" zeigt dagegen ASCII-Texte an. "sh" erzeugt folglich nichts, was "pg" sinnvoll verarbeiten könnte.

21. Zunächst wechseln Sie mit "cd /bin" in das DVZ "/bin". Beim "ls"-Befehl verwenden Sie den Schalter "-d", um zu verhindern, daß bei Dateiverzeichnissen deren Inhalt ausgegeben wird:

a) ls -d *e* .*e*
Dateien, die mit "." beginnen, werden vom Matchcode "*" nicht erfaßt. Deshalb ist diese Gruppe von Namen zusätzlich zu "*e*" zu berücksichtigen (vgl. auch die Teilaufgaben c und d).

b) ls -d c*

c) ls -d *y .*y

d) ls -d *e*m* *m*e* .*e*m* .*m*e*
Wenn mehrere Zeichen vorkommen, ist auch deren Reihenfolge relevant (z. B. erfaßt "*m*e*" zwar den Namen "acme", nicht aber "emco").

e) `ls -d [a-Z][a-Z][a-Z][a-Z]`
Jede Klammerkonstruktion steht hier für einen einzelnen Buchstaben. Bei den Bereichsangaben ist zu beachten, daß die großen Buchstaben hinter den kleinen im "Alphabet" einzuordnen sind: a, b, ... y, z, A, B, ... Y, Z.

22. `ls -d * .[!.]* ..?*`
Der Ausdruck ".[!.]*" ist wie folgt zu interpretieren: Alle Namen, die mit einem Punkt anfangen und deren zweites Zeichen kein Punkt ist; es muß jedoch mindestens ein weiteres Zeichen vorhanden sein! Das dritte Zeichen (und jedes folgende) ist beliebig, es kann auch leer sein.

Bewegen im Dateisystem

26. a) `cd /usr`
b) `cd`

28. Die Einträge, die mit "/" beginnen, sind absolute Pfadnamen, alle anderen sind relative.

29. a)

/usr1/maier/texte	texte
/usr1/maier/ordner1/kurse	ordner1/kurse
/usr1/maier/Bsp/programme	Bsp/programme
/usr1/maier2/susi	../maier2/susi
/usr1/maier2/Dokumente/briefe	../maier2/Dokumente/briefe
/usr1	..
/usr	../../usr
/usr1/maier	.

b)

briefe/susi	/usr1/maier/briefe/susi
./ottokar/rechnung	/usr1/maier/ottokar/rechnung
.	/usr1/maier
../DB	/usr1/DB
../../usr/bin/hit	/usr/bin/hit
DB/datenbank	/usr1/maier/DB/datenbank

Zugriffsrechte und Eigentümer

34. a) `chmod u-rwx .`
b) `cd ..`
Bei korrekter Ausführung der Anweisungen haben Sie sich im DVZ "Leer" "eingeschlossen". Um z. B. mit dem Befehl "cd .." - der den **relativen** Pfadnamen ".." enthält - in ein anderes DVZ wechseln zu können, muß der Kommando-Interpreter das aktuelle DVZ sozusagen lokalisieren können. Das ist ihm nun jedoch verboten, da in Teilaufgabe a) die Leseerlaubnis entzogen worden ist. (Sie könnten aus dem Dateiverzeichnis wieder "entkommen", indem Sie einen **absoluten** Pfadnamen angeben, z. B. "cd /usr1/maier/Bsp".)
c) z. B.: `chmod u+rwx /usr1/maier/Bsp/Leer`
allgemein: `chmod u+rwx $HOME/Bsp/Leer`
d) `cd ..`

Umlenkung des Standard-Ausgabe-Kanals

40. cd Bsp/Inf
ls > inhalt
ced inhalt
(Sie können nun im Editor Eintragungen vornehmen. Sie verlassen den CED mit der Taste <ENDE> und "j".)

41. b) cat adresse frech abschied > gehalt
cat adresse einladung ciao > fest
ced P.S.
(Wir nennen die Datei hier "P.S.". Schreiben Sie nun den Text des Postskriptums und verlassen Sie dann den CED wieder.)
cat P.S. >> gehalt
cat P.S. >> fest

Anlegen, Versetzen und Löschen von Dateien und DVZ

46. mv prog prog
Eine Datei kann nicht in sich selbst umbenannt werden.

mv * datei
Wenn "*" mehrere Eintragsnamen umfaßt, ist der Befehl sinnlos. Man kann nicht mehrere Dateien oder Dateiverzeichnisse zu **einem** Namen umbenennen, ebensowenig kann man Dateien bzw. Dateiverzeichnisse in eine Datei versetzen (wenn überhaupt, dann müßte das Ziel statt "datei" ein Dateiverzeichnis sein).

mv *.bak *.sik
Sinix benennt nicht jede Datei, deren Name auf ".bak" endet, in eine entsprechende Datei um, deren Name auf ".sik" endet.
Der Befehl ist nur in zwei Fällen sinnvoll:
1) *.bak und *.sik sind jeweils nur **ein** Dokument.
2) *.bak sind **nur Dateien**, und *.sik ist **ein Dateiverzeichnis**.

Wenn beide Matchcodes für mehrere Namen stehen, ist der Befehl sinnlos. Es gibt in einem Dateiverzeichnis nie mehrere Dateien oder Dateiverzeichnisse gleichen Namens. Für das Betriebssystem sind z. B. die Namen "datei1.sik" und "datei1.bak" nicht wegen des Suffix ".sik" bzw. ".bak" unterschiedlich, sondern weil die Dateinamen verschiedene Zeichen enthalten.

Suchen von Dateien und Dateiverzeichnissen

52. find . -print
Hier wurde davon ausgegangen, daß Sie sich beim Absetzen des Befehls in Ihrem Stammverzeichnis befinden. Sie können (z. B. als Benutzer "gast") auch die folgende Version eingeben:
find /usr/gast -print

53. Wechseln Sie in Ihr Stammverzeichnis (mit dem Befehl "cd" ohne Argumente"), und geben Sie folgendes ein:
a) find . -name brief -print
b) find . -name "*e*" -print

56. Im Siemens-Universum:
copy -r Bsp/Informix FIRMA

Im AT&T-Universum:
```
  mkdir FIRMA
  cd Bsp/Informix
  find . -print ¦ cpio -pd ../../FIRMA
```
Wenn man nicht vorher in das DVZ "Bsp/Informix" wechselt (man würde dann "find Bsp/Informix" schreiben), dann legt Sinix unter FIRMA nochmals die Verzeichnisse "Bsp" und darunter "Informix" an. Nach der Aufgabenstellung soll aber vom Inhalt von "Informix" unter "FIRMA" eine Sicherheitskopie erstellt werden.

Die Vergabe von Zugriffsrechten ist für beide Universen gleich und wird mit Hilfe von Matchcodes abgekürzt (hier werden allerdings nicht die Namen erfaßt, die mit einem "." beginnen!):
```
  chmod a+rwx FIRMA FIRMA/* FIRMA/*/*
```
Anmerkung: Die Zugriffsrechte werden hier auch nur drei "Ebenen" tief verändert.

58. a)
```
cd Bsp
find Informix -print ¦ cpio -ov > Sicher/firma
```

b)
```
cd
pack Sicher/firma
```

c)
```
unpack Sicher/firma.z
cd Tester
cpio -id < ../Sicher/firma
```

d)
```
cd
rm -r Tester/Informix
```

59.
```
cp prog prog
```
Eine Datei kann nicht in sich selbst kopiert werden.

```
cp alle* *
```
Da "*" zu allen Namen des aktuellen Verzeichnisses expandiert wird (z. B. auch "alle1", "alle2"), kann die Syntax nicht korrekt sein: Das zweite Argument muß ein Datei**verzeichnis** sein.

```
cp a_liste.* b_liste.*
```
Das Betriebssystem muß genau wissen, welche Dateien wohin kopiert werden sollen. Die Syntax ist nur korrekt, wenn sowohl "a_liste.*" als auch "b_liste.*" nur einen Dateinamen bezeichnen. Das ist aber nicht im allgemeinen der Fall. Eine weitere Möglichkeit ist, daß "b_liste.*" ein einzelnes Dateiverzeichnis bezeichnet, in das die mit "a_liste.*" angesprochenen Dateien kopiert werden. Auch davon kann man nicht immer ausgehen.

```
ls -R ¦ cpio -o > Ausdatei
```
Der Befehl "cpio" benötigt als Eingabe die Pfadnamen der zu verarbeitenden Dateien und Dateiverzeichnisse. "ls -R" erzeugt jedoch eine "optisch aufbereitete" Ausgabe (mit Überschriften, Leerzeilen usw.). - "cpio -o" erhält die korrekten Eingabe nur von dem Befehl "find".

```
copy -r Dateiverzeichnis Datei
```
Ein Dateiverzeichnis kann nicht in einer Datei abgelegt werden.

Diskettenbearbeitung

63. a) tar cvf /dev/fl2 Bsp/Inf

b) rm -r Bsp/Inf
tar xvf /dev/fl2 Bsp/Inf

Hintergrundprozesse und fremde Prozesse

68. cd
find . -exec ls -ld {} \; > inhalt &
Sobald der "find"-Befehl eine Datei oder ein Dateiverzeichnis gefunden hat, wird der Befehl "ls -ld" darauf angewendet. Der Ausdruck "{}" steht dabei für den gerade gefundenen Namen. Mit " \;" wird die "find"-Konstruktion abgeschlossen. Das Verarbeitungsergebnis wird in die Datei "inhalt" geschrieben. Durch das Zeichen "&" läuft das Ganze im Hintergrund ab. Etwaige Fehlermeldungen erscheinen jedoch auf dem Bildschirm!

70. ps aux ¦ fgrep root > aus-datei

Kommunikation in Sinix-Systemen

73. a) Die Adressatin heißt "karin":
ls ¦ mail karin

b) Der Adressat auf dem Rechner "MX300-1" heißt "conan":
ls ¦ mail conan@MX300-1

c) Sie müssen pro Datei eine Kommandozeile abschicken:
mail gast mgast admin < Korrespondenz/Intern/brief
mail gast mgast admin < Korrespondenz/brief
mail gast mgast admin < Korrespondenz/brief2

Automatisierung mit Hilfe von Shell-Prozeduren

78.
```
PS1="\033[07m"'?'"\033[0m"
# Mit den einfachen Anführungsstrichen wird das Fra-
# gezeichen entwertet.
export PS1
# Mit "export" wird die Variable "$PS1" auch den
# Subshells bekanntgemacht.
date
who
du -s
```

79. Wenn das erste Kommando in der Datei ".profile" der Befehl "sh" ist, zeigt Sinix direkt nach dem Login das Bereit-Zeichen ("$ ") an, es wird also eine Interpreter-Shell erzeugt. Wenn Sie dann auf <ENDE> drücken, werden die Kommandos ausgeführt, die hinter dem Befehl "sh" in der Datei ".profile" stehen.

80. Die Lösung dieser Aufgabe ist sehr schwierig, wenn man die Prozedur bis ins letzte Detail ausgestalten will. Sie kann als gelöst gelten, wenn Sie zumindestens das Prinzip der "if"- und "while"-Konstruktionen realisiert haben. Wichtig ist, daß Ihnen die Funktionsweise dieser Schleifen klar wird. Kommentare werden mit dem Zeichen "#" eingeleitet:

```
# antwort: j/n-Fragemodul zum Einbinden in
#          Shell-Prozeduren
#

antw=
# Deklaration der Variablen "$antw"

echo "
Geben Sie bitte j fuer ja ein oder
            n fuer nein: \c"
# Der "echo"-Befehl geht über 3 Zeilen,
# "\c" verhindert den Zeilenvorschub.

read antw
# Einlesen der Variablen "$antw"

while [ x"$antw" != xj -a x"$antw" != xn -a \
      x"$antw" != xJ -a x"$antw" != xN ]
# Die Konstruktion "[...]" steht für den Befehl
# "test" in seiner "eingebauten" Form. Also:
# Solange "$antw" weder j/J noch n/N ist:
do
    echo -n "
    Sie muessen j fuer ja oder
            n fuer nein eingeben: \c"
    read antw
done

if [ "$antw" = j -o "$antw" = J ]
# Je nach Antwort wird der Ende-Status besetzt:
then
    exit 0
else
    exit 1
fi
```

81. a)

```
echo "Guten Tag! Folgende Benutzer sind aktiv:"
who ¦ pg
echo "Weiter mit <RETURN>!"
read x
df
echo "Weiter mit <RETURN>!"
read x
date
echo "Weiter mit <RETURN>!"
read x
```

b) Sie haben nun die Möglichkeit, die Prozedur "antwort" aus Aufgabe 80 zu verwenden (die Datei muß vorher mit Hilfe des "chmod"-Befehls mit einer Ausführberechtigung versehen werden!):

```
echo "Wollen Sie weitermachen?"
if antwort
   then
       echo "Die Prozedur geht weiter!"
   else
       echo "Ende der Prozedur! Auf Wiedersehen!"
       exit
fi
```

Wenn Sie das nicht wollen, binden Sie das folgende Modul in die Prozedur aus Teil b) ein:

```
antw= while [ x$antw != j -a x$antw != n ]
do
    echo "Wollen Sie weitermachen? (j=ja/n=nein) \c"
done
if [ x$antw = n ]
   then
       exit
fi
```

```
c)  echo "
    Sollen die Systembelegung und Ihr Speicherplatzverbrauch ausge-
    druckt werden?"

    if antwort
    # Wieder kann die Prozedur "antwort" verwendet werden
       then

ca)        du -s $HOME > aus-datei
           # Die Variable "$HOME" enthält den Pfad Ihres
           # Stamm verzeichnisses
           df >> aus-datei
           lpr aus-datei
    fi

cb)     (du -s $HOME ; df) ¦ lpr

        echo "Die Daten werden ausgedruckt!"
    fi
```

```
d)  echo "
    Belegung des Speicherplatzes für die Kennung $USER:
    " > aus-datei
    du -s $HOME >> aus-datei

    echo "
    Auslastung der Systemkapazitäten:
    " >> aus-datei
    df >> aus-datei

    pg aus-datei

    echo "Wollen Sie diese Daten ausdrucken?"
```

```
if antwort
then
    pr aus-datei ¦ lpr
    echo "Die Daten werden gedruckt!"
fi
```

82.
```
if (who ¦ fgrep $1 > /dev/null)
# "/dev/null" ist der elektronische Papierkorb,
# es wird also verhindert, daß die Ausgabe von
# "fgrep" am Bildschirm erscheint.
   then echo "Der Benutzer $1 ist aktiv!"
   else echo "Der Benutzer $1 ist nicht aktiv!"
fi
```

83.
```
echo "Sicherung vom `date` der Kennung $USER;
Pfad des aktuellen Dateiverzeichnisses: `pwd`" > SICHER.inf
# Hier wird mit Kommandosubstitution gearbeitet.

find . -print ¦ cpio -o > SICHER.dat
pack SICHER.dat
tar cvf /dev/fl2 SICHER.inf SICHER.dat.z
```

84. Die Abfrage der Variablen "$KDNR" bleibt unverändert.

```
echo "select auftrag_nr, aufbrutto from auftrag
where kd_nr = \"$KDNR\" " > /tmp/$$.sql
# Mit $$ wird die Prozeßnummer der aktuellen Shell
# abgefragt.
isql firma /tmp/$$.sql
antw=
echo "Soll das Ergebnis gedruckt werden? (j/n) \c"
read antw
if [ x$antw = xj ]
then isql firma /tmp/$$.sql ¦ lpr
fi
```

Anhang: Kurzauflistung der Kommandos

Kurzauflistung

Kommandos, die ausschließlich in einem bestimmten Universum vorhanden sind, werden mit einem Kürzel am Anfang der jeweiligen Zeile gekennzeichnet: "att" für das AT&T-Universum und "sie" für das Siemens-Universum.
Am Ende der Zeile steht die Seitenzahl zum Nachschlagen.

Benutzerumgebung

Benutzer-Umgebung

Dateisystem

Bearbeitung des Dateisystems

Bearbeitung von Disketten und Magnetbandkassetten

Disketten / MB-Kassetten

Software-Aufrufe in Sinix

Software-Aufrufe

Kommunikation

Kommunikation in Sinix

ftp	File-Transfer-Programm aufrufen	74
hostname	aktuellen Rechnernamen ausgeben	75
mail	elektronische Post verschicken	75
mesg	Bildschirmnachrichten (nicht) empfangen	77
rcp	Verzeichniseinträge netzweit kopieren	77
rlogin	Login an einem fernen Rechner	78
rsh	Kommandos auf fernem Rechner ausführen	79
rtar	Datenträger auf fernem Rechner benutzen	80
ruptime	Zustände der Rechner im Netz ausgeben	81
rwho	aktive Sitzungen im Netz auflisten	82
talk	direkte Unterhaltung mit einem anderen Benutzer	83
write	Nachricht auf das Terminal einer	84

Systemverwaltung

Kommandos für die Systemverwaltung

chown	Eigentümer von Dateien/DVZ ändern	85
/etc/dump	Datensicherung auf Magnetbandkassette	85
/etc/fsck	Konsistenztest von Dateisystemen	88
quot -f	Dateisystem-Belegung pro Benutzer listen	89
/etc/restore	Wiedereinspielen einer "dump"-Sicherung	90
/usr/etc/secure	Sicherheitsprüfung des Systems	91
/etc/shutdown	Rechner abschalten	92
wall	(att) Bildschirmnachricht an alle aktiven Benutzer senden	93

Prozeduren

Kommandos zur Erstellung von Prozeduren

case	Verzweigung nach Wertabfragen	104
echo	Zeichenketten und Variablen ausgeben	105
exit	Shellprozedur verlassen/abbrechen	107
export	Variablen in Subshells bekanntmachen	107
for	Schleifenbildung	108

if	Verzweigung	109
read	Benutzereingabe am Bildschirm lesen	109
set	Parameter für die Shell setzen	110
shift	Stellungsvariablen verschieben	111
sleep	Prozeß vorübergehend stillegen	112
test	Bedingungen prüfen	113
[...]	Kurzform des "test"-Befehls	113
while	Schleifenkonstruktion	115

Die wichtigsten Sonderzeichen der Shell Sonderzeichen

?	Abkürzen eines Zeichens eines Eintrags	24
*	Abkürzen eines beliebigen Teils eines Eintrags	24
\	Entwertung von Sonderzeichen	30
/	Trennung der Komponenten von Pfadnamen	30
<	Umleitung der Standard-Eingabe	26
>(>)	Umleitung der Standard-Ausgabe	26
&	Erzeugung eines Hintergrund-Prozesses	27
&&	automatische Ende-Status-Auswertung (positiv)	96
¦	Datenweitergabe zwischen Kommandos ("Pipe")	25
¦¦	automatische Ende-Status-Auswertung (negativ)	96
$	bei der Verwendung von Variablen der Anfang eines Variablennamens	28
@	Abtrennung von Komponenten eines Arguments	76
#	Kommentarzeichen für Shell-Prozeduren	30
"	Zeichenketten einschließen (mit Variablensubstitution)	94
'	(Apostroph) Zeichenketten einschließen und entwerten	94
`	(Gravis) Kommandosubstitution von Zeichenketten	95
.	(Punkt) Kommandodatei starten, ohne neuen Prozeß zu erzeugen	101
=	Wertzuweisung bei Variablen und bei Schaltern	28
!	Negation bei Matchcodes	24
[]	Abkürzung von Einträgen für Bereichsangaben	24
{}	Variablennamen von anderen Zeichenketten trennen	30
()	Kommandos in einer Kommandozeile zusammenfassen	30

Register

Sinix ist das Unix der Siemens Nixdorf Informationssysteme AG. Dieses Arbeitsbuch zeigt, wie Sie dieses Betriebssystem schnell erlernen und auf einfache Weise für Ihre Aufgaben nutzen.
Nach einer schrittweisen Einführung in grundlegende Konzepte von Sinix/Unix können Sie das Erlernte in Übungsaufgaben unmittelbar ausprobieren und praktisch anwenden.
Problemlösungen aus der Praxis eines Systemverwalters runden den Lehrteil ab. Durch die alphabetisch geordnete Kommandoauflistung ist dieses Buch auch für den Profi ein nützliches Nachschlagewerk.
Beispieldaten und ausgewählte Problemlösungen finden Sie auf der beiliegenden Diskette.

R. Oldenbourg Verlag

ISBN 3-486-22004-7